



Service Level Manager Programmer's Guide

Release 2.0

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

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

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

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

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

AccessPath, AtmDirector, Browse with Me, CCDE, CCIP, CCSI, CD-PAC, *CiscoLink*, the Cisco Net*Works* logo, the Cisco *Powered* Network logo, Cisco Systems Networking Academy, the Cisco Systems Networking Academy logo, Fast Step, Follow Me Browsing, FormShare, FrameShare, GigaStack, IGX, Internet Quotient, IP/VC, iQ Breakthrough, iQ Expertise, iQ FastTrack, the iQ Logo, iQ Net Readiness Scorecard, MGX, the Networkers logo, *Packet*, RateMUX, ScriptBuilder, ScriptShare, SlideCast, SMARTnet, TransPath, Unity, Voice LAN, Wavelength Router, and WebViewer are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn, Discover All That's Possible, and Empowering the Internet Generation, are service marks of Cisco Systems, Inc.; and Aironet, ASIST, BPX, Catalyst, CCDA, CCDP, CCIE, CCNA, CCNP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, the Cisco IOS logo, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Enterprise/Solver, EtherChannel, EtherSwitch, FastHub, FastSwitch, IOS, IP/TV, LightStream, MICA, Network Registrar, PIX, Post-Routing, Pre-Routing, Registrar, StrataView Plus, Stratm, SwitchProbe, TeleRouter, and VCO are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries.

All other trademarks mentioned in this document or Web site are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0105R)

Service Level Manager Programmer's Guide
Copyright © 2000-2001, Cisco Systems, Inc.
All rights reserved.

Table Of Contents

Chapter 1 Introduction.....	9
Data Model	9
SLM Architecture	9
Support for Cisco Interface Specs and APIs	10
Cisco Developer Support Web Sites.....	11
Chapter 2 Getting Started.....	12
CiscoWorks2000 Authentication and Authorization Overview	12
Authenticating with CiscoWorks2000	12
Chapter 3 Communicating with the SLM Server.....	14
Result Formats.....	15
Chapter 4 Folders and Folder Lists	16
Servlet API for Folders	16
Servlet URL.....	16
Servlet Parameters	16
FolderList and Folder Schema.....	18
SlcList and Slc as Part of a Folder.....	18
Chapter 5 SLCLists.....	20
Servlet API for SLCLists.....	20
Servlet URL.....	20
Servlet Parameters	20
SLCList Schema.....	21
Chapter 6 SLCs	22
Servlet API for SLCs	22
Servlet URL.....	22
Servlet Parameters	22
ApplyTime	25
Service Level Agreement (SLA)	26
ICMPMetric	27
ICMPDeviceSpec	27
ICMPThreshold.....	28
UDPMetric.....	29
UDPDeviceSpec	29
UDPThreshold	30
DNSMetric.....	31
DNSDeviceSpec	31

DNSThreshold	32
HTTPMetric	33
HTTPDeviceSpec	34
HTTPThreshold	34
VoIPMetric.....	35
VoIPDeviceSpec	36
VoIPThreshold.....	37
TCPMetric	37
TCPDeviceSpec	38
TCPThreshold	39
DHCPMetric	39
DHCPDeviceSpec.....	40
DHCPThreshold.....	41
SLC Block Diagram	42
Chapter 7 Inventory Services	43
Servlet API for Inventory	43
Servlet URL.....	43
Servlet Parameters	43
DeviceList Schema	44
SAADeviceList Schema	45
Chapter 8 Data Export Services	46
Servlet API for Data Export	46
Servlet URL.....	46
Servlet Parameters	46
Export Result Schema	49
ICMPResult, UDPResult, DNSResult, TCPResult, DHCPResult.....	50
HTTPResult.....	54
VoIPResult	57
Chapter 9 Threshold Alert Servlet	60
Servlet API for Threshold Alert	60
Servlet URL.....	60
Servlet Parameters	60
ThresholdAlert Schema.....	60
Chapter 10 Report Navigator Cross-Launch	63
Chapter 11 View Collection Manager Status	64
Chapter 12 Putting It All Together	65
Using Cookies	65

Getting Device Lists.....	68
Device List Request	68
Device List Response.....	69
Defining SLCs and SLAs.....	69
Request to Add a New SLC	69
Response to Add Request	69
Getting Existing SLC Handles and Names	69
Using SLC Lists	69
Using Folder Lists	70
Getting SLA Handles and Names from an Existing SLC.....	71
SLC Request	71
SLC Response	71
Retrieving SLC/SLA Data.....	72
SLA Data Request	72
SLA Data Response.....	73
Appendix A XML Document Type Definitions (DTDs)	74
FolderList.dtd	74
SLCList.dtd	74
SLC.dtd	75
UDPMetric.dtd.....	76
ICMPMetric.dtd	77
DNSMetric.dtd.....	77
HTTPMetric.dtd.....	78
VoIPMetric.dtd.....	78
TCPMetric.dtd	79
DHCPMetric.dtd	79
SlamError.dtd	80
SlamAdminResult.dtd	80
SlmResults.dtd:	80
ICMPResult.dtd:	81
UDPResult.dtd:	81
DNSResult.dtd:	81
TCPResult.dtd:	82
DHCPResult.dtd:	82
HTTPResult.dtd:	82
VoIPResult.dtd:	83
DeviceList.dtd	84

SAADeviceList.dtd..... 84

ViewList.dtd 84

ThresholdAlert.dtd 84

Appendix B SLM Error Code Format 86

Appendix C SLM Troubleshooting Guide..... 87

Appendix D Base64 Encoding Example 89

Definitions

<i>Availability</i>	The percentage of time that a device or link is operational.
<i>CM</i>	SLM Collection Manager
<i>DNS</i>	Domain Name Service.
<i>DTD</i>	Document Type Definition. Syntax used to define an XML schema.
<i>IOS</i>	Cisco Internetworking Operating System.
<i>Jitter</i>	Variability between packet delivery times, used as a measure of voice data quality.
<i>Latency</i>	The amount of time it takes for packets to travel from one device to another.
<i>MTBF</i>	Mean Time Between Failure. This is the average length of time that a device or link is operational.
<i>MTTR</i>	Mean Time To Repair. This is the average length of time that a device or link is NOT operational.
<i>RME</i>	Resource Manager Essentials
<i>SAA</i>	Service Assurance Agent. An IOS feature that measures Round Trip Response time (RTR). SAA provides different probes for measuring various types of response (for example, HTTP, DNS, Echo, Jitter).
<i>SAA Device</i>	A network device that supports the SAA feature set.
<i>SLM / SLAM</i>	Service Level Manager.
<i>SLA</i>	Service Level Agreement. The specific metric and threshold to which a service provided has agreed to meet.
<i>SLC</i>	Service Level Contract. The agreed-upon <i>set</i> of SLAs that the service providers have agreed to meet with their customers.
<i>SNMP</i>	Simple Network Management Protocol.
<i>TOS</i>	Type of Service. A special field in IP packet headers that indicates the route type (for example, speed versus cost) to use when forwarding the packet. See RFC 1349
<i>URL-encoded</i>	Strings that are passed as data via HTTP URLs must be encoded to prevent any embedded special characters from being interpreted as part of the URL. This means that spaces are replaced by the "+" character, and special characters such as "&" are replaced by % followed by the hexadecimal equivalent of the ASCII character. For example, the string <i>Jack & Jill</i> would become <i>Jack+%26+Jill</i> .
<i>XML</i>	Extensible Markup Language.

Chapter 1 Introduction

This guide describes the services and features of the Service Level Manager (SLM) that can be programmatically controlled. The provided interfaces allow third-party application vendors or enterprise customers to integrate SLM into their own network management tools. At a high level, SLM provides the following services:

- A platform independent, open standards API built upon HTTP and XML.
- An interface to create/modify/delete Service Level Contracts (SLCs) and Service Level Agreements (SLAs).
- An interface to retrieve the data gathered as part of an SLA.
- An interface to display a report for a given SLC or SLA.
- An interface to retrieve the SLA information associated with an SNMP trap that was generated by an SAA operation that exceeded its threshold.

Data Model

The SLAs defined in SLM are based upon the following high-level objects:

- **Service Level Contract:** This represents the overall contract between a service provider and a customer. An SLC contains one or more SLAs and defines the time range over which its SLAs apply (for example, Monday – Friday, 9:00am – 5:00pm). Each SLC has a name that must be unique on the SLM server.
- **Service Level Agreement:** An SLA encapsulates the type of metric that should be monitored (for example, DNS response time), the thresholds for the given metric, and the list of device pairs covered by the SLA. **Note:** More than one device pair can be covered under a single SLA, and the same device pair may be covered under more than one SLA.
- **Metrics:** This is the type of test that should be performed or monitored. In SLM 2.0 it is possible to define Round Trip Response (ICMP and UDP) metrics, DNS metrics, HTTP metrics, TCPConnect metrics, DHCP metrics, and Voice-over-IP (VoIP) metrics. Each type of metric has its own unique set of thresholds. For example, the response time metric has thresholds for latency and availability.
- **Folder:** This is a user interface convention in SLM that allows users to organize their SLCs into different folders. Folders could also be useful for third-party application vendors who want to store SLCs that they define in a standard folder (perhaps for branding purposes).

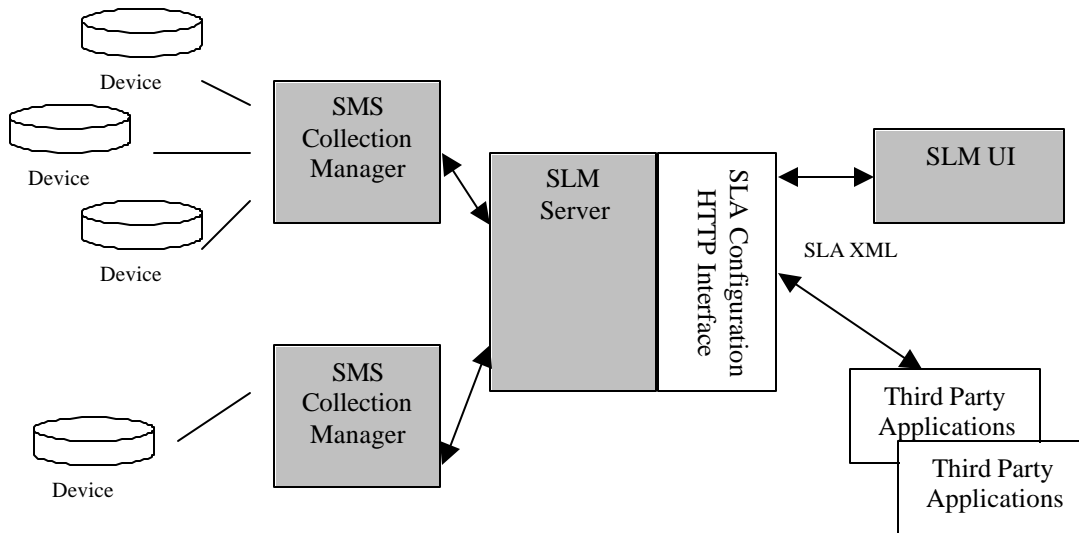
The chapters that follow present details on communicating with the SLM server and all of the above elements and their attributes. Authentication for communicating with the SLM server is accomplished through the use of cookies. See the “Getting Started” chapter for more information.

SLM Architecture

SLM can be broken down into three major subsystems:

- **XML Interface:** SLM exposes interfaces for working with SLCs and retrieving results through the use of XML. Clients connect to the SLM server using HTTP and pass requests (for example, Create SLC) as a series of HTTP POST parameters. The SLM server will service the request and return any results (also in XML).
- **SLM Server:** The SLM server archives and processes SLC requests (for example, create/modify SLC). When an SLC is created, the SLM server contacts the appropriate SMS Collection Manager (CM) to gather the data. It periodically sweeps the CMs to gather the data and store the results in the database.

- SLM Collection Manager (CM):** SLM works in conjunction with the CM, a software bundle designed to perform data collection and aggregation functions for network management applications. It is used by SLM to collect performance data needed to monitor SLAs. The SLM server ships with an embedded CM as part of the Service Management Solution (SMS). Additional remote CMs can be installed on any remote device that meets CM server requirements. This distributed data collection and aggregation system provides a cost-effective, scalable solution. The SLM server sends requests to the CMs to gather the data necessary for each SLA. The CMs will then perform all of the required setup/polling of the managed devices (for example, SAA device).



The use of HTTP and XML eliminates common problems incumbent in other technologies such as CORBA or RMI (for example, Firewall issues, language independence etc.). This allows applications to be developed using any language, such as Perl, Python, Java, or C++, on virtually any operating system, and to be distributed across the network.

Support for Cisco Interface Specs and APIs

Cisco has a new support program for developers who are enabling products with Cisco-supported interfaces. The Developer Support Program is being developed to provide formalized support for Cisco interfaces to accelerate the delivery of compatible solutions to Cisco customers.

The Developer Support Program offers the following benefits:

- Minimal support fees
- Flexible support model—purchase support as needed or for a period of time
- Consistent level of support—defined problem priority and escalation guidelines
- Deliver products to market faster—dedicated program with interface experts to assist you

Cisco Developer Support Web Sites

To find out more about this program and obtain the Developer Support Agreement, visit our web site at:

<http://www.cisco.com/go/developersupport>

Upon receipt of your signed agreement, we will send you your contract ID number and instructions for opening support cases with Cisco Developer Support Engineers.

Cisco Developer Support

developer-support@cisco.com

<http://www.cisco.com/go/developersupport>

Chapter 2 Getting Started

The SLM server uses the CiscoWorks2000 security and authentication framework, which is based on session cookies, to validate remote requests. Third-party applications need to log in and be authenticated by the SLM server before accessing SLM services. The CiscoWorks2000 authentication and authorization framework are presented below.

CiscoWorks2000 Authentication and Authorization Overview

By default, CiscoWorks2000 performs authentication based on a username and password pair. When you log in to CiscoWorks2000, you enter this username and password pair and the pair is compared against entries in the CiscoWorks2000 database. If the pair exists and is correct, the login process is successful and you get a session cookie back from the server.

Authorization however, is performed based on roles. CiscoWorks2000 has seven different roles. They are:

- Help Desk
- Approver
- Network Operator
- Network Administrator
- System Administrator

Each of these roles has different level of privileges on the system. In addition, users can have multiple roles associated with their usernames. Therefore, depending on the roles, you might not be able to execute some operations on the system. For example, a user with a help desk role will not be able to add users into the system.

To add a new user into the system, you need to log in with Administrator privileges and select on Server Configuration -> Security -> Add Users from the CiscoWorks2000 navigation tree.

Authenticating with CiscoWorks2000

Before communicating with the SLM server through the XML interface, you need to authenticate with CiscoWorks2000. This is accomplished by calling the CiscoWorks2000 authentication servlet.

Authentication Servlet URL

http://SLM_SERVER:PORT/CSCOnm/servlet/com.cisco.nm.cmf.servlet.CsAuthServlet

where SLM_SERVER is the hostname of the machine on which SLM is installed, and PORT is the port number (the default port number is 1741) of the machine on which the SLM is installed.

Servlet Parameters

The table below outlines the servlet parameters used to log in to and log out from the SLM server.

Request		Response
Command	Parameters	Returned Value
cmd=authUser	name= <i>userName</i> pwd= <i>password</i>	true:X ,where X is the privileges code or false
cmd=logout		True

The password needs to be encoded using BASE64 encoding. For more information about BASE64 encoding, see RFC1521. Sample code for generating BASE64 encoded strings is included in Appendix D.

For example, to log in to the SLM server called "MyServer" with the user name "admin" and password "admin" (encoded as YWRtaW4=), the following HTTP request needs to be sent:

http://myserver:1741/CSCOnm/servlet/com.cisco.nm.cmf.servlet.CsAuthServlet?cmd=authUser&name=admin&pwd=YWRtaW4=

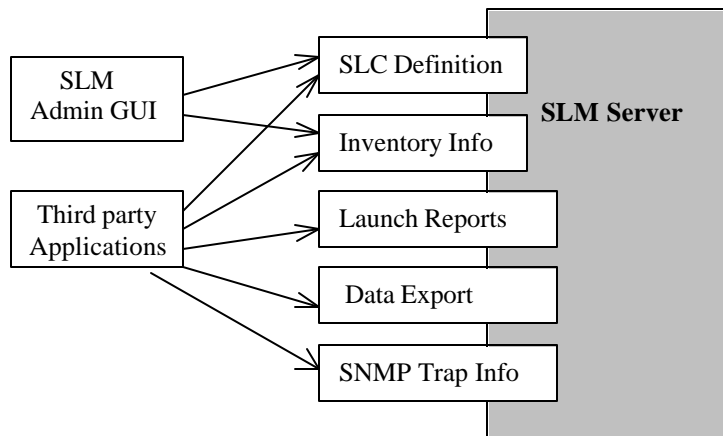
If the authentication is successful, the servlet returns a session cookie, which will be used to validate subsequent requests made by the application. If the application runs within a browser (for example, Java applet) then the session cookie will be sent with the HTTP requests by the browser. Otherwise, the application must explicitly include the session cookie in the HTTP requests. An authenticated session remains valid until a logout request is made or a timeout occurs.

Note:

- It could take a few seconds for the session cookie authorization to complete before further commands can be issued. See Chapter 12 for sample code that can be used to communicate with the servlet.
- SLM XML interface requires users to have a network administrator or a system administrator role to access the interface. Therefore, when you authenticate with CiscoWorks2000, make sure that you are using a username with the correct roles. Otherwise, you will not be authorized to call the SLM XML interface.

Chapter 3 Communicating with the SLM Server

Applications communicate with the SLM server via a set of Java servlets that expose XML interfaces. For example, the *AdminServlet* provides an open interface to access and manipulate SLM objects such as SLCs and folders.



In general, communication with the server involves the following:

- Authenticate with the CiscoWorks2000 Server and obtain session ID.
- Determine to the URL of the appropriate servlet.
- Set -the session cookie in the HTTP request.
- Specify the type or class of object you want to work with (for example, folder, SLC).
- Specify the type of operation (for example, Add, Modify, Enumerate).
- Provide operation specific data (for example, SLC XML file).

All of the parameters are passed using HTTP POST. Results will be returned back in an operation-specific XML object.

Each request comprises two parts: operation parameters and data (optional). The operation parameters consist of the following key-value pairs, encoded as CGI parameters of the POST request:

- Class (Folder, SLC, Inventory)
- Operation (Enumerate, Get, Modify, Add, Delete, Move)
- Parameters (operation's parameters)

The data part is an XML buffer and is encoded in the last key-value pair of the POST request. Some requests have only the operation portion and do not require the data portion. The XML buffers must conform to the Document Type Definitions (DTDs) outlined in this document. Strings

that are passed as data via HTTP URLs must be encoded to prevent any embedded special characters from being interpreted as part of the URL. This means that spaces are replaced by the “+” character, and special characters such as “&” are replaced by “%” followed by the hexadecimal equivalent of the ASCII character. For example, the string Jack & Jill would become Jack+%26+Jill. See RFC 1349 for more information on URL-encoded strings.

Result Formats

Depending on the type of request and its execution status, the following objects are returned in the response:

Result Object	Description
FolderList	An XML buffer that conforms to the FolderList DTD and contains a list of the current Folder names and the names of their associated SLCs. See Chapter 4.
SLCList	An XML buffer that conforms to the SLCList DTD and contains a list of all current SLCs in the system. See Chapter 5.
SLC	An XML buffer that conforms to the SLC DTD and contains detailed information about the requested SLC. See Chapter 6.
SlamAdminResult	A simple XML buffer that conforms to the SlamAdminResult DTD. This type of result is used to return simple request specific objects and has the form: <pre><SlamAdminResult> returned_value </SlamAdminResult></pre>
SlamError	An XML buffer that conforms to the SlamError DTD. This result is used to flag errors in carrying out the requested operation. See Appendix B

Chapter 4 Folders and Folder Lists

Folders are a user interface convention in SLM that allows users to organize their SLCs. Folders are also useful for third-party application vendors who want to store SLCs in a standard folder (perhaps for branding purposes). Folder services are based upon the following objects:

- **FolderList:** A list of the folders that currently exist on the SLM server.
- **Folder:** A Folder has a name and a list of the names of the SLCs contained in the folder.

The following sections will cover the supported operations for folders, the servlet API for operating on folders and the object attributes of folders. The formal DTDs for folders will be presented at the end of the chapter. SLC operations will be addressed separately in the next chapter.

Servlet API for Folders

Servlet URL

`http://SLM_SERVER:PORT/CSCOnm/servlet/com.cisco.nm.slam.admin.servlet.AdminServlet`

where "SLM_SERVER" is the hostname and "PORT" is the port number (default port number is 1741) of the machine on which the SLM server is installed.

For example:

`http://slmserver:1741/CSCOnm/servlet/com.cisco.nm.slam.admin.servlet.
AdminServlet?class=Folder&cmd=enumerate`

Note: For authentication, a valid session cookie must be included in the HTTP requests to this servlet. See Chapter 2 for details about session authentication.

Servlet Parameters

The table below outlines the folder operations and parameters supported by the Admin Servlet. The request's POST parameters would be structured as follows:

class= Folder & cmd=command & param=value

Folder Operations		
REQUEST		RESPONSE
Command	Parameters	Return Value (if successful)
Enumerate		<p>An XML buffer that conforms to the FolderList DTD and contains a list of the current folder names and the names and handles of SLCs contained the folders. A FolderList containing two folders would have the following form:</p> <pre> <FolderList> <Folder Handle=1 > <Name>Default</Name> ...Sls list of Folder 1... </Folder> </Folder Handle=2 > <Name>TestFolder</Name> ...SlcList of Folder 2... </Folder> </FolderList> </pre>
Add	name = <i>folderName</i>	<p>The handle of the new folder is returned:</p> <pre> <SlamAdminResult> "folderID " </SlamAdminResult> </pre>
Delete	folderhandle= <i>folderID</i>	<pre> <SlamAdminResult> "Success" </SlamAdminResult> </pre> <p>Note: A folder can not be deleted unless it is empty. For example, the following error will be returned if the folder is not empty:</p> <pre> <SlamError Code="SLAM_ANI_UNABLE_TO_DELETE_FOLDER" Sev="SEV"> <Descr>Could not delete Folder ABC, it is in use.</Desc> </SlamError> </pre>
Get	folderhandle= <i>folderID</i>	An XML buffer that conforms to the FolderList DTD and contains only the requested folder handle and its associated SLCs.
Modify	folderhandle = <i>folderID</i> name = <i>folderName</i>	<p>A command that changes the name of the folder referenced by the <i>folderID</i> handle. On success, the following result is returned:</p> <pre> <SlamAdminResult> "Success" </SlamAdminResult> </pre>

MoveSLC	folderhandle = <i>folderID</i> slchandle = <i>SLC_ID</i>	A command that moves an SLC to the folder referenced by the <i>folderID</i> handle on the SLM administration server. On success, the following result is returned: <SlamAdminResult> "Success" </SlamAdminResult>
---------	---	---

FolderList and Folder Schema

A FolderList contains a list of the folders that are defined on the SLM server and the list of SLCs that are defined in each folder. A folder list has the following components:

FolderList (Folder)

Folder (Name, Handle, SlcList)

Component	Description	Required	Quantity
FolderList	An XML list of one or more folder objects.	Yes	1
Folder	An XML object used to encapsulate a group of SLCs. There is always at least one folder defined on the SLM server, although it might be empty.	Yes	1 or more
Name	The name of the folder. The name must be unique, and cannot exceed 20 characters in length.	Yes	1
Handle	An integer value used as an internal identifier for the folder. The number is assigned by the SLM server. Clients must use this handle when performing other operations such as folder deletion. This is used for the "id" Command parameter when operating on a folder.	Yes	1
SlcList	The list of SLCs contained in the folder. If the folder contains no SLCs, the SLCList will be empty. SLCLists are described below.	Yes	1

SlcList and Slc as Part of a Folder

An SlcList contains a list of the names and handles of the SLCs that are stored in a particular folder. An SlcList has the following components:

SlcList (Slc)

Slc (Name, LastModifiedTime, Handle)

Component	Description	Required	Quantity
Slc	An SLC contained in the SLC List.	No	0 or more
Name	The name of the SLC. The name must be unique on the SLM server and cannot exceed 40 characters in length.	Yes	1
LastModifiedTime	A date stamp showing when the SLC was last changed. This is a string of the form dd-MMM-	Yes	1

	yyyy hh:mm:ss ZZZ, for example, 08-Sep-1999 15:03:02GMT.		
Handle	An integer value used as an internal identifier for the SLC. The SLM server assigns this number. Clients must use this handle when performing other operations by specifying this handle in the "handle" command parameter.	Yes	1

Chapter 5 SLCLists

It is not necessary to access SLCs via the folder and FolderList mechanisms. The SLCList command allows retrieval of basic information about all SLCs that are defined on the SLM server.

Servlet API for SLCLists

Servlet URL

`http://SLM_SERVER:PORT/CSCOnm/servlet/com.cisco.nm.slam.admin.servlet.AdminServlet`

where "SLM_SERVER" is the hostname and "PORT" is the port number (default port number is 1741) of the machine on which the SLM server is installed.

Note: For authentication, a valid session cookie must be included in the HTTP requests to this servlet. See Chapter 2 for details about session authentication.

Servlet Parameters

The table below outlines the folder operations and parameters supported by the Admin servlet. The request's POST parameters would be structured as follows:

class= SLC & cmd=command

Currently, the only supported command on SLC Lists is the Enumerate command, described below.

Folder Operations		
REQUEST		RESPONSE
Command	Parameters	Return Value (if successful)
Enumerate		<p>An XML buffer that conforms to the SLCList DTD and contains a list of the current SLC names and handles. A SLCList containing two SLCs would have the following form:</p> <pre> <SLCList> <SLC Handle=9876 > <Name> "My SLC Name" </Name> <LastModifiedTime> "08-Sep-1999 15:03:01 GMT" </LastModifiedTime> </SLC> </SLC Handle=1234 > <Name> "Another SLC Name" </Name> <LastModifiedTime> "28-Sep-1999 15:14:02 GMT" </LastModifiedTime> </SLC> </SLCList> </pre>

SLCList Schema

An SLCList contains a list of the SLCs that are defined on the SLM server. An SLCList has the following components:

SLCList (SLC)

SLC (Name, Handle, LastModifiedTime)

Component	Description	Required	Quantity
SLC	A high-level XML description of an SLC defined on the SLM server.	No	0 or more
Name	The name of the SLC	Yes, if SLC is defined	1
Handle	An integer value used as an internal identifier for the SLC. The SLM server assigns this number.	Yes, if SLC is defined	1
LastModifiedTime	Date stamp showing when the SLC was last modified. This is a string of the form dd-MMM-yyyy hh:mm:ss ZZZ, for example, 08-Sep-1999 15:03:02 GMT.	Yes, if SLC is defined	1

Chapter 6 SLCs

This chapter introduces how SLCs and SLAs are modeled in SLM. At a high level, an SLC is modeled around the following objects:

- **Service Level Contract (SLC):** This represents the overall contract between a service provider and a customer. A SLC contains one or more SLAs and defines the time range over which its SLAs apply (for example, Monday – Friday, 9:00am – 5:00pm). Each SLC has a name that must be unique on the SLM server.
- **Service Level Agreement (SLA):** A SLA encapsulates the type of metric that should be monitored (for example, DNS response time), the thresholds for the given metric, and the list of device pairs covered by the SLA. : More than one device pair can be covered under a single SLA.
- **Metrics:** This is the type of test that should be performed or monitored. In SLM 2.0 it is possible to define ICMP metrics, UDP metrics, DNS metrics, HTTP metrics, TCP Connect metrics, DHCP metrics, and VoIP metrics. Each type of metric has its own unique set of thresholds.
- **Thresholds:** Each type of metric has its own unique set of possible threshold values. For example, the ICMP metric has thresholds for latency and availability.
- **DeviceSpec:** A device specification defines the pair of devices between which an SLA has been established. For each type of metric, there is a matching DeviceSpec entity. For example, an HTTP metric has an associated HTTPDeviceSpec to define the source device and the destination HTTP server.

Note: When defining device pairs it is strongly recommended that the source device is retrieved from the inventory service. See Chapter 7.

The following sections present details on all of the above components. Once all of the components have been covered, the formal DTDs are presented.

Servlet API for SLCs

Servlet URL

http://SLM_SERVER:PORT/CSCOnm/servlet/com.cisco.nm.slam.admin.servlet.AdminServlet

where “SLM_SERVER” is the hostname and “PORT” is the port number (default port number is 1741) of the machine on which the SLM server is installed.

Note: For authentication, a valid session cookie must be included in the HTTP requests to this servlet. See Chapter 3 for details about session authentication.

Servlet Parameters

The table below outlines the SLC operations and parameters supported by the Admin servlet. The request's POST parameters would be structured as follows:

class=SLC&cmd=command¶m=value&data="URL-encoded XML data string..."

The data value, if given, must be URL-encoded.

See Chapter 12 for an example of an SLC Request and Response.

SLC Operations			
REQUEST			RESPONSE
Command	Parameters	Data (XML)	Return Value (if successful)
Enumerate			An XML list of all the SLC names and handles defined on the system. See Chapter 5 for a description.
Add	folder = <i>folderID</i> (optional; will be created in default folder if not specified)	New SLC-SLA XML data string. Must be URL encoded.	<SlamAdminResult> <i>SLC_ID</i> </SlamAdminResult>
Delete	slchandle = <i>SLC_ID</i>		Deletes the SLC with a handle of <i>SLC_ID</i> <SlamAdminResult> "Success" </SlamAdminResult>
Get	slchandle = <i>SLC_ID</i>		SCL-SLA XML data string of the form: <Slc Enabled="True" Handle="1234"> <Name>Test SLC</Name> <Comment>This is an SLC comment</Comment> <ApplyTime> <i>...described elsewhere...</i> </ApplyTime> <Sla> <i>...described elsewhere...</i> </Sla> <Sla> <i>...described elsewhere...</i> </Sla> <LastModifiedTime> 23-Jun-1999 22:17:35 GMT </LastModifiedTime> </Slc>
Modify	slchandle = <i>SLC_ID</i>	New SLC-SLA XML data string	<SlamAdminResult> "Success" </SlamAdminResult>

Note: SLC and SLA handles are required for the modify operation.

SLC Schema

A SLC represents contract between a service provider and a customer. An SLC contains one or more SLAs and defines the time range over which its SLAs apply (for example, Monday – Friday, 9:00 am – 5:00 pm). A SLC has the following components:

Slc (Name, Comment, Enabled, Sla, ApplyTime, Handle, LastModifiedTime)

Component	Description	Required	Quantity
Name	The name of the SLC. The name must be unique on the SLM server and cannot exceed forty (40) characters in length	Yes	1
Comment	A description that can be associated with the SLC. The comment cannot exceed 255 characters in length.	No	0 or 1
Enabled	Flag indicating if the SLC is enabled or not. If an SLC is disabled, the data collection will be stopped for all of the SLAs defined in the SLC. Allowed values are "True" or "False". The default value is "True".	No	0 or 1
Sla	The list of service level agreements associated with the SLC.	Yes	1 or more
ApplyTime	The days of the week and the time over which the SLC (and underlying SLAs) apply. For example, you can define the SLC to apply Monday through Friday, from 9:00 am to 5:00 pm	Yes	1
Handle	An integer value used as an internal identifier for the SLC. The SLM server assigns this number. It is not a required parameter. Normally the SLM server uses the SLC <i>Name</i> to identify a SLC. However, it will use the ID if present in order to improve performance. The Handle can also be used to update the SLC Name during a Modify operation.	No	0 or 1
LastModified Time	Date stamp showing when the SLC was added or last changed on the SLM server. This is an optional parameter that will be filled in by the SLM server after the SLC is created on the server. This is a string of the form dd-MMM-yyyy hh:mm:ss ZZZ, for example, 08-Sep-1999 15:03:02 GMT.	No	0 or 1

ApplyTime

The ApplyTime component defines the days of the week, and time over which the SLC (and underlying SLAs) apply. Each day of the week has a separate time range. For example, you can define the SLC to apply Monday through Friday, from 6:00 am to 11:00 pm, and 9:00 am to 5:00 pm on Saturdays and Sundays. The time zone can be specified as either Local (for time relative to the SLM server), or UTC for universal time (also known as GMT). Only Local time zone is supported in the current release. ApplyTime has the following components:

ApplyTime (ApplyMon, ApplyTue, ApplyWed, ApplyThu, ApplyFri, ApplySat, ApplySun)

ApplyMon (FromTime, ToTime) – similar syntax for other days of the week

Component	Description	Required	Quantity
Zone	Defines the time zone in which the apply-times are defined. The possible values are LOCAL or UTC. LOCAL refers to the local time on the SLM server. UTC is universal time (also known as GMT). Only LOCAL time zone is supported in the current release.	Yes	1
Apply<Day>	Contains the time range for which the SLC applies for the given day of the week (for example, "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", and "Sun").	Yes	1 or more
FromTime	The start time that the SLC applies on each day. This must be a whole number between 0 and 24. The default is "00". Note:: If both the FromTime and ToTime are equal to zero ("00"), then the SLC does not apply to the entire day.	Yes	1
ToTime	The ending time that the SLC applies on each day. This must be a whole number between 0 and 24 and must be greater than the FromTime. The default is "24."	Yes	1

An ApplyTime element might look like:

```
<ApplyTime Zone=LOCAL>
<ApplyMon>
<FromTime>00</FromTime>
<ToTime>23</ToTime>
</ApplyMon>
<ApplyTue>
<FromTime>00</FromTime>
<ToTime>23</ToTime>
</ApplyTue>
...
<ApplySun>
<FromTime>08</FromTime>
<ToTime>17</ToTime>
</ApplySun>
</ApplyTime>
```

Service Level Agreement (SLA)

A SLA encapsulates the type of metric that should be monitored (for example, DNS response time), the thresholds for the given metric, and the list of device pairs covered by the SLA. Note that more than one device pair can be covered under a single SLA. A SLA has the following components:

Sla (Name, Comment, CreateTime, (UDPMetric | ICMPMetric | HTTPMetric | DNSMetric | VoIPMetric|TCPMetric|DHCPMetric), Handle)

Component	Description	Required	Quantity
Name	The name of the SLA. The name must be unique within the parent SLC, and cannot exceed forty (40) characters in length	Yes	1
Comment	A description that can be associated with the SLA. The comment cannot exceed 255 characters in length.	No	0 or 1
CreateTime	The time that the SLA was added to the SLM server. This is an optional parameter that will be filled in by the SLM server, not client applications. This is a string of the form dd-MMM-yyyy hh:mm:ss ZZZ, for example, 08-Sep-1999 15:03:02 GMT.	No	0 or 1
<*>Metric	<p>The type of metric that the SLA will monitor. SLM 2.0 supports the following metrics:</p> <ul style="list-style-type: none"> • UDPMetric • ICMPMetric • HTTPMetric • DNSMetric • VoIPMetric • TCPMetric • DHCPMetric <p>Details on each metric and its associated components are presented in the sections that follow.</p>	Yes	1
Handle	An integer value used as an internal identifier for the SLA. The SLM server assigns this number. It is not a required parameter. Normally the SLM server uses the SLA <i>Name</i> to identify an SLA. However, it will use the ID if present in order to improve performance.	No	0 or 1

An SLA might look like:

```
<SLA>
<Name>ICMP SLA</Name>
<Comment>This is an SLA comment </Comment>
```

```
<CreateTime>23-Jun-1999 22:17:35 GMT </CreateTime>
<ICMPMetric>...described elsewhere... </ICMPMetric>
</SLA>
```

ICMPMetric

An ICMPMetric is used primarily to measure the network latency between two devices. In addition to latency, this metric allows users to monitor the availability, Mean-Time-Between-Failures (MTBF), and Mean-Time-To-Repair (MTTR) of the path between the two devices. The ICMPMetric has the following components:

ICMPMetric (ICMPDeviceSpec, SamplingInterval, TOS, ICMPThreshold)

Component	Description	Required	Quantity
ICMPDeviceSpec	The list of device pairs for which round trip response should be measured. See the ICMPDeviceSpec section for details.	Yes	1 or more
SamplingInterval	The interval that an ICMP probe should be sent, in minutes. Allowed values are "1", "5", "10", "15", or "30". The default is 5.	No	0 or 1
PayloadSize	The number of bytes to include in the data section of the ICMP packet. Must be between 28 and 16348 octets; the default is 64.	No	0 or 1
TOS	The Type of Service value for the ICMP probe packet. This can be a number from 0 to 63. The default is 0.	No	0 or 1
ICMPThreshold	The threshold values for the ICMPMetric. See the ICMPThreshold section for details.	Yes	1

An ICMPMetric might look like:

```
<ICMPMetric SamplingInterval="15">
<ICMPDeviceSpec>...described elsewhere... </ICMPDeviceSpec>
<ICMPDeviceSpec>...described elsewhere... </ICMPDeviceSpec>
<ICMPThreshold>...described elsewhere... </ICMPThreshold>
</ICMPMetric>
```

ICMPDeviceSpec

An ICMPDeviceSpec (device specification) is used to define the source and destination devices in an ICMPMetric. The source device must support SAA 2.1.0. The ICMPDeviceSpec has the following components:

ICMPDeviceSpec (SourceDevice, TargetDevice, CreateTime)

Component	Description	Required	Quantity
SourceDevice	The SAA-capable device that will send a probe to the target device. The SourceDevice you select must already exist in the Essentials Inventory database on the SLM server.	Yes	1
TargetDevice	The fully qualified domain name of the target device.	Yes	1
CreateTime	The time that the device pair was added to the SLA. The SLM server will fill in this field. Its purpose is to help the reporting framework know how much historical data is available for the device. This is a string of the form dd-MMM-yyyy hh:mm:ss ZZZ, for example, 08-Sep-1999 15:03:02 GMT.	No	0 or 1

An ICMPDeviceSpec might look like:

```
<ICMPDeviceSpec>
<SourceDevice> source.domain.com </SourceDevice>
<TargetDevice>target.domain.com</TargetDevice>
</ICMPDeviceSpec>
```

ICMPThreshold

The ICMPThreshold defines the set of threshold values for the ICMPMetric. The ICMPThreshold has the following components:

ICMPThreshold (AvgHourlyLatency, AvgDailyLatency, DailyAvailability, MonthlyAvailability)

Component	Description	Required	Quantity
AvgHourlyLatency	The threshold value for the average hourly latency. The number should be a whole number representing latency in milliseconds (for example, 60, 1000).	Yes	1
AvgDailyLatency	The threshold value for the average daily latency. The number should be a whole number representing latency in milliseconds (for example, 60, 1000).	Yes	1
DailyAvailability	The expected availability percentage for any given day. The number should be a floating point number (for example, 99.9 or 92.50). The possible range is 0.00 to 100.00.	Yes	1
MonthlyAvailability	The expected availability percentage for any given month. The number should be a floating point number (for example, 99.9 or 92.50). The possible range is 0.00 to 100.00.	Yes	1

An ICMPThreshold might look like:

```
<ICMPThreshold>
<AvgHourlyLatency>90</AvgHourlyLatency>
<AvgDailyLatency>75</AvgDailyLatency>
<DailyAvailability>99.5</DailyAvailability>
<MonthlyAvailability>99.8</MonthlyAvailability>
</ICMPThreshold>
```

UDPMetric

A UDPMetric is used primarily to measure the UDP latency between two devices. It differs from ICMPMetric in that it uses UDP rather than ICMP. This metric also allows users to monitor the availability, MTBF, and MTTR of the path between the two devices. The UDPMetric has the following components:

UDPMetric (UDPDeviceSpec, SamplingInterval, PayloadSize, TOS, UDPThreshold)

Component	Description	Required	Quantity
UDPDeviceSpec	The list of device pairs for which round trip response should be measured. See the UDPDeviceSpec section for details.	Yes	1 or more
SamplingInterval	The interval that a UDP probe should be sent, in minutes. Allowed values are "1", "5", "10", "15", and "30". The default is 5.	No	0 or 1
PayloadSize	The size (in bytes) of the UDP probe packet's data section. Must be between 4 and 1464 octets; the default is 64.	No	0 or 1
Port	The port number on the target device to which the packets will be sent. The default is 7 (UDP Echo).	No	0 or 1
TOS	The Type of Service value for the UDP probe packet. This can be a number from 0 to 63. The default is 0.	No	0 or 1
UDPThreshold	The threshold values for the UDPMetric. See the UDPThreshold section for further details.	Yes	1

A UDPMetric might look like:

```
<UDPMetric SamplingInterval="15" PayloadSize="256">
<UDPDeviceSpec>...described elsewhere... </UDPDeviceSpec>
<UDPDeviceSpec>...described elsewhere... </UDPDeviceSpec>
<UDPThreshold>...described elsewhere... </UDPThreshold>
</UDPMetric>
```

UDPDeviceSpec

A UDPDeviceSpec (device specification) defines the source and destination devices in a UDPMetric. The source device must support SAA 2.1.0. The UDPDeviceSpec has the following components:

UDPDeviceSpec (SourceDevice, TargetDevice, CreateTime)

Component	Description	Required	Quantity
SourceDevice	The SAA-capable device that will send a probe to the target device. The SourceDevice must already exist in the Essentials Inventory database and support SAA.	Yes	1
TargetDevice	The fully qualified domain name of the target device.	Yes	1
CreateTime	The time that the device pair was added to the SLA. The SLM server will fill in this field. Its purpose is to help the reporting framework know how much historical data is available for the device. This is a string of the form dd-MMM-yyyy hh:mm:ss ZZZ, for example, 08-Sep-1999 15:03:02 GMT.	No	0 or 1

A UDPDeviceSpec might look like:

```
<UDPDeviceSpec>
<SourceDevice> source.domain.com </SourceDevice>
<TargetDevice>target.domain.com</TargetDevice>
<CreateTime> 10-25-1999 03:22:19 GMT </CreateTime>
</UDPDeviceSpec>
```

UDPThreshold

The UDPThreshold defines the set of threshold values for the UDPMetric. The UDPThreshold has the following components:

UDPThreshold (HourlyLatency, DailyLatency, HourlyAvailability, DailyAvailability)

Component	Description	Required	Quantity
AvgHourlyLatency	The threshold value for the average hourly latency. The number should be a whole number representing latency, in milliseconds (for example, 60, 1000).	Yes	1
AvgDailyLatency	The threshold value for the average daily latency. The number should be a whole number representing latency, in milliseconds (for example, 60, 1000).	Yes	1
DailyAvailability	The expected availability percentage for any given day. The number should be a floating point number (for example, 99.9 or 92.50). The possible range is 0.00 to 100.00	Yes	1
MonthlyAvailability	The expected availability percentage for any given month. The number should be a floating point number (for example, 99.9 or 92.505). The possible range is 0.00 to 100.00	Yes	1

A UDPThreshold might look like:

```
<UDPThreshold>
<AvgHourlyLatency>90</AvgHourlyLatency>
<AvgDailyLatency>75</AvgDailyLatency>
<DailyAvailability>99.5</DailyAvailability>
<MonthlyAvailability>99.8</MonthlyAvailability>
</UDPThreshold>
```

DNSMetric

The DNSMetric measures the response time of a DNS server. A DNS request is sent from the SAA device to a specified DNS server to resolve the user-specified IP address. This metric also allows users to monitor the availability, MTBF, and MTTR of the DNS server. The DNSMetric has the following components:

DNSMetric (DNSDeviceSpec, SamplingInterval, TestIPAddr, DNSThreshold)

Component	Description	Required	Quantity
DNSDeviceSpec	The list of device pairs (SAA source and a DNS server) and sample IP address for which DNS lookup responsetime should be measured. See the DNSDeviceSpec section for details.	Yes	1 or more
SamplingInterval	The interval during which a DNS probe should be sent, in minutes. Allowed values are "1", "5", "10", "15", and "30". The default is 5.	No	0 or 1
TestIPAddr	The address to use in the DNS reverse name lookup test. If not included, the address of the SLM server host will be used.	No	0 or 1
DNSThreshold	The threshold values for the DNSMetric. See the DNSThreshold section for further details.	Yes	1

A DNSMetric might look like:

```
<DNSMetric SamplingInterval="30">
<DNSDeviceSpec>...described elsewhere...</DNSDeviceSpec>
<DNSDeviceSpec>...described elsewhere...</DNSDeviceSpec>
<TestIPAddr>131.108.234.17</TestIPAddr>
<DNSThreshold>...described elsewhere...</DNSThreshold>
</DNSMetric>
```

DNSDeviceSpec

A DNSDeviceSpec is used to define the source device and the DNS server for use in a DNSMetric. The source device must support SAA 2.1.0. The DNSDeviceSpec has the following components:

DNSDeviceSpec (SourceDevice, TargetDevice, CreateTime)

Component	Description	Required	Quantity
SourceDevice	The SAA-capable-device that will send a DNS probe to the specified DNS server. The DNSSourceDevice must already exist in the Essentials Inventory database and support SAA.	Yes	1
TargetDevice	The fully qualified domain name of the target DNS server.	Yes	1
CreateTime	The time that the device pair was added to the SLA. The SLM server will fill in this field. Its purpose is to help the reporting framework know how much historical data is available for the device. This is a string of the form dd-MMM-yyyy hh:mm:ss ZZZ, for example, 08-Sep-1999 15:03:02 GMT.	No	0 or 1

A DNSDeviceSpec might look like:

```
<DNSDeviceSpec>
<SourceDevice> source.domain.com </SourceDevice>
<TargetDevice> target.domain.com </TargetDevice>
</DNSDeviceSpec>
```

DNSThreshold

The DNSThreshold defines the set of threshold values for the DNSMetric. The DNSThreshold has the following components:

DNSThreshold (AvgHourlyLatency, AvgDailyLatency, DailyAvailability, MonthlyAvailability)

Component	Description	Required	Quantity
AvgHourlyLatency	The threshold value for the average hourly latency. The number should be a whole number representing latency, in milliseconds (for example, 60, 1000).	Yes	1
AvgDailyLatency	The threshold value for the average daily latency. The number should be a whole number representing latency, in milliseconds (for example, 60, 1000).	Yes	1
DailyAvailability	The expected availability percentage for any given day. The number should be a floating point number (for example, 99.9 or 92.50). The possible range is 0.00 to 100.00	Yes	1
MonthlyAvailability	The expected availability percentage for any given month. The number should be a floating point number (for example, 99.9 or 92.50). The possible range is 0.00 to 100.00	Yes	1

A DNSThreshold might look like:

```
<DNSThreshold>
```

```

<AvgHourlyLatency>90</AvgHourlyLatency>
<AvgDailyLatency>75</AvgDailyLatency>
<DailyAvailability>99.5</DailyAvailability>
<MonthlyAvailability>99.8</MonthlyAvailability>
</DNSThreshold>

```

HTTPMetric

The HTTPMetric is used to measure the response time of a web server. An HTTP request is sent from the SAA device to a specified web server to retrieve a user-specified URL. The HTTPMetric has the following components:

HTTPMetric (HTTPDeviceSpec, SamplingInterval, URLPath, ProxyServer, Port, TOS, Timeout, NameServer, CacheEnable, HTTPThreshold)

Component	Description	Req'd	Quantity
HTTPDeviceSpec	The list of device pairs (SAA source and an HTTP server), for which an HTTP GET should be performed. See the HTTPDeviceSpec section for details.	Yes	1 or more
SamplingInterval	The interval during which an HTTP probe should be sent, in minutes. Allowed values are "1", "5", "10", "15", and "30". The default is 5.	No	0 or 1
URLPath	An optional string to append to the URL of the destination web server name. The complete URL is built as follows: <code>http://<TargetDevice>:<Port>/<URLPath></code> . For example, if the TargetDevice is <code>www.dom.com</code> , the Port is "100", and the URLPath is "mydir/index.html", the HTTP probe will be applied to: HTTP://www.dom.com:100/mydir/index.html . If the element is blank, then no URL path is added.	No	0 or 1
Port	The port number of the web server. The default is 80.	No	0 or 1
ProxyServer	The hostname of the proxy server to be used for the target device. If the element is blank, then no proxy server is used.	No	0 or 1
NameServer	The name server that the source device should use to resolve the HTTP server's hostname. If the element is blank, then the default name server is used.	No	0 or 1
TOS	The Type of Service (priority) value for the HTTP probe packet. This can be a number from 0 to 63	No	0 or 1
Timeout	The HTTP server timeout value, in milliseconds. The default is 5000 milliseconds.	No	0 or 1
CacheEnable	Defines whether the HTTP server should use its cache. Possible values are "True" or "False". The default is "False".	No	0 or 1
HTTPThreshold	Defines the threshold values for the HTTPMetric. See the HTTPThreshold section for details.	Yes	1

An HTTPMetric might look like:

```
<HTTPMetric>
<HTTPDeviceSpec>...described elsewhere... </HTTPDeviceSpec>
<URLPath> "Mktg/CustomerData/Index.html" </URLPath>
<SamplingInterval>15</SamplingInterval>
<HTTPThreshold>...described elsewhere... </HTTPThreshold>
</HTTPMetric>
```

HTTPDeviceSpec

An HTTPDeviceSpec is used to define the source device and the hostname or IP address of the target HTTP server for use in a HTTPMetric. The source device must support SAA 2.0. The HTTPDeviceSpec has the following components:

HTTPDeviceSpec (SourceDevice, TargetDevice, CreateTime)

Component	Description	Required	Quantity
SourceDevice	The router that will send out the HTTP tests. This is the SAA-capable device that will send an HTTP probe to the specified web server.	Yes	1
TargetDevice	Fully qualified name of the HTTP server address to test (for example, www.domain.com).	Yes	1
CreateTime	The time that the device pair was added to the SLA. The SLM server will fill in this field. Its purpose is to help the reporting framework know how much historical data is available for the device. This is a string of the form dd-MMM-yyyy hh:mm:ss ZZZ, for example, 08-Sep-1999 15:03:02 GMT.	No	0 or 1

An HTTPDeviceSpec might look like:

```
<HTTPDeviceSpec>
<SourceDevice> source.domain.com </SourceDevice>
<TargetDevice> target.domain.com </TargetDevice>
<CreateTime> 10-25-1999 03:22:19 GMT </CreateTime>
</HTTPDeviceSpec>
```

HTTPThreshold

The HTTPThreshold defines the set of threshold values for the HTTPMetric. The HTTPThreshold has the following components:

HTTPThreshold (AvgHourlyLatency, AvgDailyLatency)

Component	Description	Required	Quantity
AvgHourlyLatency	The threshold value for the average hourly latency. The number should be a whole number	Yes	1

	representing latency, in milliseconds (for example, 60, 1000).		
AvgDailyLatency	The threshold value for the average daily latency. The number should be a whole number representing latency, in milliseconds (for example, 60, 1000).	Yes	1

An HTTPThreshold might look like:

```
<HTTPThreshold>
<AvgHourlyLatency>90</AvgHourlyLatency>
<AvgDailyLatency>75</AvgDailyLatency>
</HTTPThreshold>
```

VoIPMetric

The VoIPMetric is used to measure the jitter between two SAA devices. The VoIPMetric has the following components:

VoIPMetric (VoIPDeviceSpec, SamplingInterval, PacketsPerSample, PayloadSize, InterPacketInterval, EnableControl, TOS, SourcePort, TargetPort, VoIPThreshold)

Component	Description	Required	Quantity
VoIPDeviceSpec	The list of SAA device pairs between which jitter should be measured. See the VoIPDeviceSpec section for details.	Yes	1 or more
SamplingInterval	The interval during which the jitter probes should be sent, in minutes. Allowed values are "1", "5", "10", "15", and "30". The default is 5.	No	0 or 1
PacketsPerSample	The number of packets to send each interval. The default is 10.	Yes	0 or 1
PayloadSize	The size of the data portion of each packet (in octets). This must be between 16 and 1500 octets. The default is 64.	No	0 or 1
InterPacketInterval	The time, in milliseconds, between each packet. (This is called Inter-Packet Delay in the GUI.) This value must be between 1 and 60000. The default is 20. InterPacketInterval must not exceed the specified SamplingInterval.	Yes	0 or 1
EnableControl	Allows SAA control packets to enable the desired listening port on the target SAA device. If the port has already been enabled via the SAA control, this is not required. The default is "True."	No	0 or 1
TOS	The Type of Service value for the VoIP probe packet. This can be a number from 0 to 63.	No	0 or 1
SourcePort	The port number of the source device from which to send the packets. This should not be a well-	Yes	0 or 1

	known port. The typical address range used by SLM is 8020-8050. The default is 8020.		
TargetPort	The Target Device port number to which to send the packets. This should not be a well-known port. The typical address range used by SLM is 8020-8050. The default is 8020.	Yes	0 or 1
VoIPThreshold	The threshold values for the VoIPMetric. See the VoIPThreshold section for details.	Yes	1

A VoIPMetric might look like:

```
<VoIPMetric>
<VoIPDeviceSpec>...described elsewhere...</VoIPDeviceSpec>
<VoIPDeviceSpec>...described elsewhere...</VoIPDeviceSpec>
<SamplingInterval>15</SamplingInterval>
<PacketsPerSample>10</PacketsPerSample>
<PayloadSize>1500</PayloadSize>
<InterPacketInterval>10</InterPacketInterval>
<VoIPThreshold>...described elsewhere...</VoIPThreshold>
</VoIPMetric>
```

VoIPDeviceSpec

A VoIPDeviceSpec defines the source and destination SAA devices for use in a VoIPMetric. Both the source and destination devices must support SAA 2.1.0 and be managed by the Essentials Inventory. The VoIPDeviceSpec has the following components:

VoIPDeviceSpec (SourceDevice, TargetDevice, CreateTime)

Component	Description	Required	Quantity
SourceDevice	The SAA-capable device that will send the jitter probes to the destination device. The device must already exist in the Essentials Inventory database.	Yes	1
TargetDevice	Fully qualified domain name of the target device (for example, myhost.domain.com).	Yes	1
CreateTime	The time that the device pair was added to the SLA. The SLM server will fill in this field. Its purpose is to help the reporting framework know how much historical data is available for the device. This is a string of the form dd-MMM-yyyy hh:mm:ss ZZZ, for example, 08-Sep-1999 15:03:02 GMT.	No	0 or 1

A VoIPDeviceSpec might look like:

```
<VoIPDeviceSpec>
<SourceDevice> source.domain.com </SourceDevice>
<TargetDevice> target.domain.com</TargetDevice>
</VoIPDeviceSpec>
```

VoIPThreshold

The VoIPThreshold defines the set of threshold values for the VoIPMetric. The VoIPThreshold has the following components:

VoIPThreshold (Jitter, RoundTripLatency, PacketLoss)

Component	Description	Required	Quantity
Jitter	The threshold for positive and (absolute) negative jitter, in milliseconds. The default is 75. This is a positive integer.	No	0 or 1
RoundTripLatency	The threshold value for the average hourly latency. The number should be a whole number representing latency, in milliseconds (for example, 60, 1000).	Yes	1
PacketLoss	Maximum packet loss as a percentage of traffic, averaged over 1 hour. The default is 2.50%. The number should be a floating point number (for example, 1.9 or 2.50). The valid range is 0.00 to 100.00.	No	0 or 1

A VoIPThreshold might look like this:

```
<VoIPThreshold>
<Jitter>60</Jitter>
<RoundTripLatency>100</RoundTripLatency>
<PacketLoss>2.50</PacketLoss>
</VoIPThreshold>
```

TCPMetric

A TCPMetric is used primarily to measure the TCP connection time between two devices. This metric also allows users to monitor the availability, MTBF, and MTTR of the path between the two devices. The TCPMetric has the following components:

TCPMetric (TCPDeviceSpec, SamplingInterval, PayloadSize, Port, ResponderEnable, TOS, TCPThreshold)

Component	Description	Required	Quantity
TCPDeviceSpec	The list of device pairs for which TCP connection time should be measured. See the TCPDeviceSpec section for details.	Yes	1 or more
SamplingInterval	The interval during which a TCPConnect probe should be sent, in minutes. Allowed values are "1", "5", "10", "15", and "30". The default is 5..	No	0 or 1
Port	The port number on the target device to which the connection will be made. The default is 23.	No	0 or 1

ResponderEnable	Uses the responder feature. The default is false. Setting this attribute to "true" requires that the responder be enabled in the target device. See Cisco IOS documents for enabling the responder.	No	0 or 1
TOS	The Type of Service value for the TCP connection. This can be a number from 0 to 63. The default is 0.	No	0 or 1
TCPThreshold	The threshold values for the TCPMetric. See the TCPThreshold section for further details.	Yes	1

A TCPMetric might look like:

```
<TCPMetric SamplingInterval="15" PayloadSize="256">
<TCPDeviceSpec>...described elsewhere... </TCPDeviceSpec>
<TCPDeviceSpec>...described elsewhere... </TCPDeviceSpec>
<TCPThreshold>...described elsewhere... </TCPThreshold>
</UDPMetric>
```

TCPDeviceSpec

A TCPDeviceSpec (device specification) defines the source and destination devices in a TCPMetric. The source device must support SAA 2.1.0. The TCPDeviceSpec has the following components:

TCPDeviceSpec (SourceDevice, TargetDevice, CreateTime)

Component	Description	Required	Quantity
SourceDevice	The SAA-capable device that will send a probe to the target device. The SourceDevice must already exist in the Essentials Inventory database and must support SAA.	Yes	1
TargetDevice	The fully qualified domain name of the target device.	Yes	1
CreateTime	The time that the device pair was added to the SLA. The SLM server will fill in this field. Its purpose is to help the reporting framework know how much historical data is available for the device. This is a string of the form dd-MMM-yyyy hh:mm:ss ZZZ, for example, 08-Sep-1999 15:03:02 GMT.	No	0 or 1

A TCPDeviceSpec might look like:

```
<TCPDeviceSpec>
<SourceDevice> source.domain.com </SourceDevice>
<TargetDevice>target.domain.com</TargetDevice>
<CreateTime> 10-25-1999 03:22:19 GMT </CreateTime>
</TCPDeviceSpec>
```

TCPThreshold

The TCPThreshold defines the set of threshold values for the TCPMetric. The TCPThreshold has the following components:

TCPThreshold (AvgHourlyLatency, AvgDailyLatency, DailyAvailability, MonthlyAvailability)

Component	Description	Required	Quantity
AvgHourlyLatency	The threshold value for the average hourly latency. The number should be a whole number representing latency, in milliseconds (for example, 60, 1000).	Yes	1
AvgDailyLatency	The threshold value for the average daily latency. The number should be a whole number representing latency, in milliseconds (for example, 60, 1000).	Yes	1
DailyAvailability	The expected availability percentage for any given day. The number should be a floating point number (for example, 99.9 or 92.50). The possible range is 0.00 to 100.00.	Yes	1
MonthlyAvailability	The expected availability percentage for any given month. The number should be a floating point number (for example, 99.9 or 92.50). The possible range is 0.00 to 100.00.	Yes	1

A TCPThreshold might look like:

```
<TCPThreshold>
<AvgHourlyLatency>90</AvgHourlyLatency>
<AvgDailyLatency>75</AveDailyLatency>
<DailyAvailability>99.5</DailyAvailability>
<MonthlyAvailability>99.8</MonthlyAvailability>
</TCPThreshold>
```

DHCPMetric

A DHCPMetric is used primarily to measure the DHCP service response time. This metric also allows users to monitor the availability, MTBF, and MTTR of the service. The DHCPMetric has the following components:

DHCPMetric (DHCPDeviceSpec, SamplingInterval, DHCPThreshold)

Component	Description	Required	Quantity
DHCPDeviceSpec	The list of device pairs for which DHCP response time should be measured. See the DHCPDeviceSpec section for details.	Yes	1 or more
SamplingInterval	The interval during which a DHCP probe should	No	0 or 1

	be sent in minutes. Allowed values are "1", "5", "10", "15", and "30". The default is 5.		
DHCPThreshold	The threshold values for the DHCPMetric. See the DHCPThreshold section for further details.	Yes	1

A DHCPMetric might look like:

```
<DHCPMetric SamplingInterval="15" Broadcast="true">
<DHCPDeviceSpec>...described elsewhere... </DHCPDeviceSpec>
<DHCPDeviceSpec>...described elsewhere... </DHCPDeviceSpec>
<DHCPThreshold>...described elsewhere... </DHCPThreshold>
</DHCPMetric>
```

DHCPDeviceSpec

A DHCPDeviceSpec (device specification) defines the source and target devices in a DHCPMetric. The source device must support SAA 2.1. The target device is the DHCP server if applicable. For broadcast probes, this component is ignored. For SLM 2.0, only broadcast DHCP probes are supported. The DHCPDeviceSpec has the following components:

DHCPDeviceSpec (SourceDevice, TargetDevice, CreateTime)

Component	Description	Required	Quantity
SourceDevice	The SAA-capable device that will send a DHCP request to the target device. The SourceDevice must already exist in the Essentials Inventory database and support SAA 2.1.	Yes	1
TargetDevice	The target DHCP server. For SLM 2.0, the only accepted value is 255.255.255.255 (broadcast probes.)	Yes	1
CreateTime	The time that the device pair was added to the SLA. The SLM server will fill in this field. Its purpose is to help the reporting framework know how much historical data is available for the device. This is a string of the form dd-MMM-yyyy hh:mm:ss ZZZ, for example, 08-Sep-1999 15:03:02 GMT.	No	0 or 1

A DHCPDeviceSpec might look like:

```
<DHCPDeviceSpec>
<SourceDevice> source.domain.com </SourceDevice>
<TargetDevice> BROADCAST</TargetDevice>
<CreateTime> 10-25-1999 03:22:19 GMT </CreateTime>
</DHCPDeviceSpec>
```

DHCPThreshold

The DHCPThreshold defines the set of threshold values for the DHCPMetric. The DHCPThreshold has the following components:

DHCPThreshold (AvgHourlyLatency, AvgDailyLatency, DailyAvailability, MonthlyAvailability)

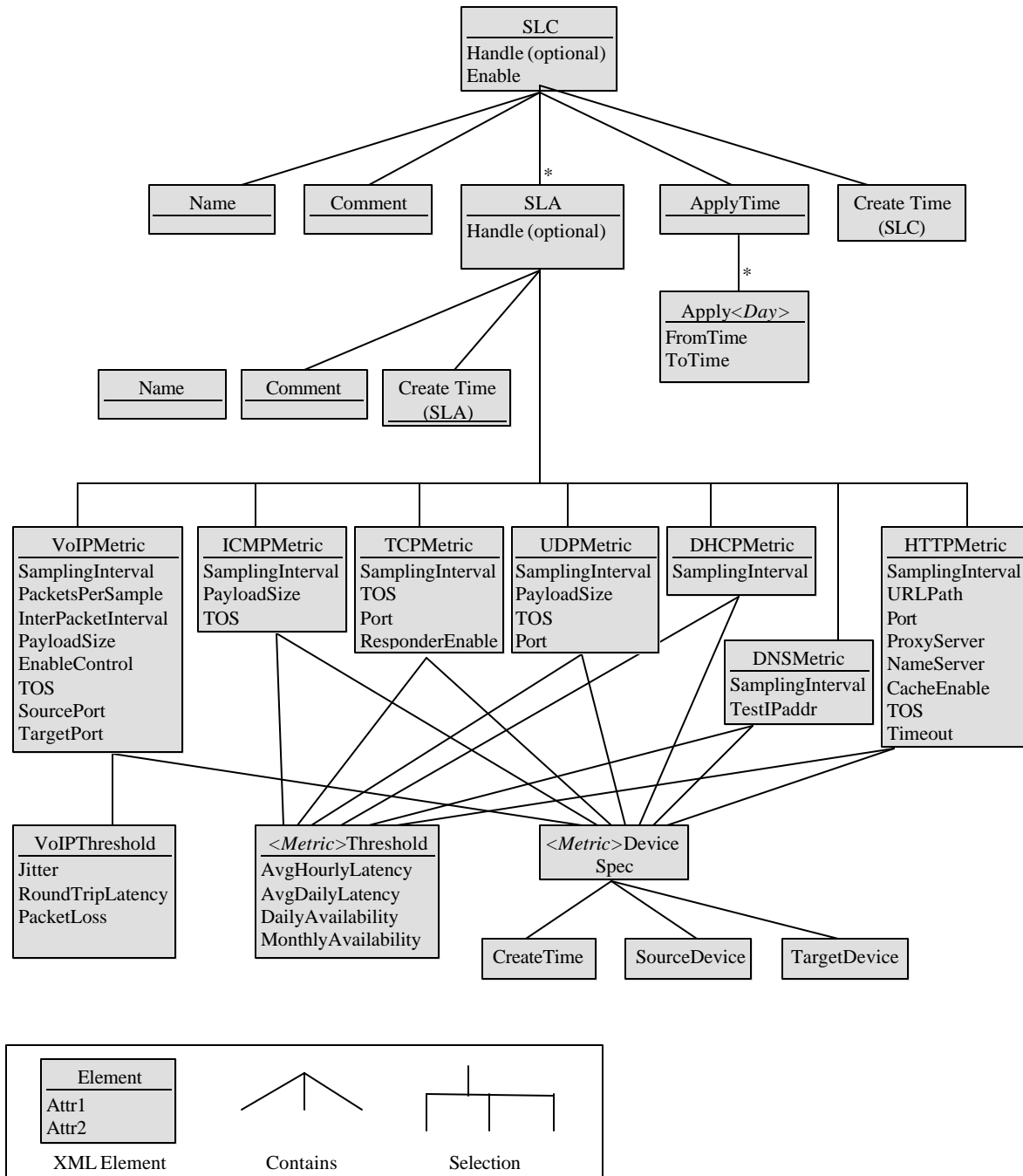
Component	Description	Required	Quantity
AvgHourlyLatency	The threshold value for the average hourly latency. The number should be a whole number representing latency in milliseconds (for example, 60, 1000).	Yes	1
AvgDailyLatency	The threshold value for the average daily latency. The number should be a whole number representing latency, in milliseconds (for example, 60, 1000).	Yes	1
DailyAvailability	The expected DHCP service availability percentage for any given day. The number should be a floating point number (for example, 99.9 or 92.50). The possible range is 0.00 to 100.00.	Yes	1
MonthlyAvailability	The expected DHCP service availability percentage for any given month. The number should be a floating point number (for example, 99.9 or 92.50). The possible range is 0.00 to 100.00.	Yes	1

A DHCPThreshold might look like:

```
<DHCPThreshold>
<AvgHourlyLatency>90</AvgHourlyLatency>
<AvgDailyLatency>75</AveDailyLatency>
<DailyAvailability>99.5</DailyAvailability>
<MonthlyAvailability>99.8</MonthlyAvailability>
</DHCPThreshold>
```

SLC Block Diagram

The figure below illustrates how the various components that define an SLC are related.



Chapter 7 Inventory Services

The Inventory API can be used to retrieve a list of devices that are being managed by Essentials and are therefore available for use by SLM. Inventory services are accessed via the “*Inventory*” request class. Use this interface to determine if particular devices are before you create SLAs for those devices.

Note: It is *strongly* recommended that you create SLCs using the fully qualified device names retrieved from the inventory service.

Servlet API for Inventory

Servlet URL

`http://SLM_SERVER:PORT/CSCOnm/servlet/com.cisco.nm.slam.admin.servlet.AdminServlet`

where “SLM_SERVER” is the hostname and “PORT” is the port number (default port number is 1741) of the machine on which the SLM server is installed.

Servlet Parameters

The table below shows the folder operations and parameters supported by the Admin Servlet. The request's POST parameters would be structured as follows:

class= *Inventory* & cmd= *Command* & Parameter= *Value*

Inventory Operations		
REQUEST		RESPONSE
Command	Parameters	Return Value (if successful)
GetDevices	<p>Type=Device_Type The possible types are listed in the Device Type table below. If omitted, the names of all devices managed by Essentials will be returned.</p> <p>View = <View_ID> You can also retrieve devices based on a view ID. Use the GetViews command to retrieve the available views.</p> <p>Specifying both Type and View will result in the returned list containing devices that match both values. (the intersection of the two lists).</p>	<p>An XML buffer that conforms to the DeviceList DTD and contains a list of the device names. A DeviceList containing two devices would have the following form:</p> <pre><DeviceList type=Device_Type > <Device> Fully_Qualified_Hostname1 </ > <Device> Fully_Qualified_Hostname2 </ > </DeviceList></pre>

GetSAADevices	View = <View_ID> You can also retrieve devices based on a view ID. Use the GetViews command to retrieve the available views.	<p>An XML buffer that conforms to the SAADeviceList DTD and contains a list of the device names and the supported SAA probe type(s). Only devices that support SAA 2.1.0 are returned. A DeviceList containing two devices would have the following form:</p> <pre> <SAADeviceList> <SAADevice types="ICMP,UDP,DNS,HTTP" > Hostname1</ > <SAADevice types="ICMP,UDP,DNS,HTTP" > Hostname2</ > </SAADeviceList> </pre>
GetViews	None	<p>An XML buffer that conforms to the DeviceViewList DTD and contains a list of the static and dynamic device views in the Essentials Inventory. This is used to retrieve just those devices that match some predefined set of criteria.</p> <pre> <DeviceViewList> <View Id= 1> "All Catalysts" </View> <View Id=2> "All 4000 Routers </View> <View Id=3> "All North American Devices" </View> <View Id=4> "All Japan Devices" </View> </DeviceViewList> </pre>

DeviceList Schema

A DeviceList contains a list of all the devices that are managed by the Essentials Inventory and match the query. A DeviceList has the following components:

DeviceList (Type, View, Device)

Component	Description	Required	Quantity
Type	The types of devices listed. The possible types are described in the Device Type table below.	Yes	1
View	A view ID, if such an ID was used to generate the DeviceList.	No	0 or 1
Device	The fully qualified name of a device. If the hostname was not entered into the Essentials Inventory, or if the device has no hostname defined, the IP address will be used. If no devices match the type, this list might be empty.	No	0 or more

Device_Type can be any of the following:

Device Type	Description
ICMP	Source devices that support ICMP Probe.
UDP	Source devices that support UDP Probe.
DNS	Source devices that support DNS Probe.
HTTP	Source devices that support HTTP Probe.
VoIP	Source/Target devices that support Jitter Probe.
TCP	Source devices that support TCPConnect Probe.
DHCP	Source devices that support DHCP Probe.
All	All managed devices in inventory.

SAADeviceList Schema

A SAADeviceList contains a list of all the SAA devices that are managed by the Essentials Inventory and match the query. A DeviceList has the following components:

SAADeviceList (View, SAADevice)

SAADevice(Types)

Component	Description	Required	Quantity
View	A view ID, if such an ID was used to generate the SAADeviceList.	No	0 or 1
SAADevice	The fully qualified name of an SAA device. If the hostname was not entered into the Essentials Inventory, or if the device has no hostname defined, the IP address will be used. If no devices match the type, this list might be empty.	No	0 or more
Types	The CSV list of SAA probe type(s) supported by the SAA device. See the Device Type table above for the list of SAA probe types.	Yes	1 or more per device

Chapter 8 Data Export Services

The Data Export API can be used to retrieve data from the SLM server for existing SLCs and SLAs. The SLC handle (slcid) and SLA handle (slaid) are returned when an SLC or SLA is created. Alternatively, you can use the folder API to retrieve the SLC identifier and then use that handle to get the specific SLC XML, which will contain the component SLAs and their IDs. The SLM Data Export can be accessed by calling SlamDataExportServlet. You must set a username and password in the authentication cookie prior to using any SLM servlet.

Servlet API for Data Export

Servlet URL

`http://SLM_SERVER:PORT/CSCOnm/servlet/com.cisco.nm.slam.report.servlet.SlamDataExportServlet`

where "SLM_SERVER" is the hostname and "PORT" is the port number (the default port number is 1741) of the machine on which the SLM server is installed.

Servlet Parameters

The table below shows the export operations and parameters supported by the SlamDataExportServlet. The request's POST parameters would be structured as follows:

**slcid=slc_id&slaid=sla_id&srcdev=source_device &destdev=destination_device
&start=start_date &end=end_date & type=type_string**

Export Operation Parameters		
Request Parameter	Description	Required
slcid	Numeric SLC Identifier.	Yes
slaid	Numeric SLA Identifier.	Yes
srcdev	The fully qualified name of the source device. If not specified, data for all source devices in the SLA is returned.	No
destdev	The fully qualified name of the destination device. If not specified, data for all destination devices in the SLA is returned.	No
start	The start time and date for the portion of the table to export. The format is dd-MMM-yyyy:hh:mm For example, <i>08-Sep-1999:15:03</i> Data from the table recorded on or after that date and time will be exported. If not specified, the start date will be the earliest date at which data was collected. The time is relative to UTC.	No
end	The end time and date for the portion of the table to export. The format is the same as for the start time. Data recorded from the start time up to before the end time will be exported. If the end time is omitted, all data from the start time to the latest data available in the table is exported, subject to practical file size limitations. The time is relative to UTC.	No

type	The type of data to be exported. Possible values are shown in the Type table below. If omitted, 5-minute values for all types applicable to the SLA will be returned except that for HTTP and VoIP, the Hourly value will be returned. If the Type value selected in the request does not match the type of the requested SLA, an error is returned.	No
------	--	----

The following table shows the possible Type field values:

Type Name	Description
DNSLatencyMin	Latency values for DNS probes. This type will return the 5-minute values or the smallest available time interval if probes were collected at greater than 5-minute intervals. If more than one probe was collected in a 5-minute period, the Minimum, Maximum, and Average values are returned. Otherwise, the value is returned as Average.
DNSLatencyHourly	Latency values for DNS probes. Includes Min, Max, and Average values for each 1-hour time period.
ICMPLatencyMin	Latency values for ICMP probes. This type will return the 5-minute values or the smallest available time interval if probes were collected at greater than 5-minute intervals. . If more than one probe was collected in a 5-minute period, the Minimum, Maximum, and Average values are returned. Otherwise, the value is returned as Average.
ICMPLatencyHourly	Latency values for ICMP probes. Includes Min, Max, and Average values for each 1-hour time period.
HTTPLatencyHourly	Latency values for HTTP probes. Includes Min, Max, and Average values for each 1-hour time period. HTTP results include the time taken for DNS lookup, TCP connect, and the HTTP transaction as separate values.
VoIPHourly	1-hour values for VoIP probes. This includes Minimum, Maximum, and Average values for each of the data types returned in a Jitter probe.
UDPLatencyMin	Latency values for UDP probes. This type will return the 5-minute values or the smallest available time interval if probes were collected at greater than 5-minute intervals. If more than one probe was collected in a 5-minute period, the Minimum, Maximum, and Average values are returned. Otherwise, the value is returned as the Average.
UDPLatencyHourly	Latency values for UDP probes. Includes Min, Max, and Average values for each 1-hour time period.
TCPLatencyMin	Latency values for TCPConnect probes. This type will return the 5-minute values or the smallest available time interval if probes were collected at greater than 5-minute intervals. If more than one probe was collected in a 5-minute period, the Minimum, Maximum, and Average values are returned. Otherwise, the value is returned as the Average.
TCPLatencyHourly	Latency values for TCPConnect probes. Includes Min, Max, and Average values for each 1-hour time period.
DHCPatencyMin	Latency values for DHCP probes. This type will return the 5-minute values, or the smallest available time interval if probes were collected

	at greater than 5-minute intervals. If more than one probe was collected in a 5-minute period, the Minimum, Maximum, and Average values are returned. Otherwise, the value is returned as the Average.
DHCP Latency Hourly	Latency values for DHCP probes. Includes Min, Max, and Average values for each 1-hour time period.
DNS Availability Hourly	Availability values for DNS probes. Includes Device Down and Link Down values for each 1-hour time period, expressed as a decimal value from 0.0 to 100.0 (0% to 100%) downtime per period.
ICMP Availability Hourly	Availability values for ICMP probes. Includes Device Down and Link Down values for each 1-hour time period, expressed as a decimal value from 0.0 to 100.0 (0% to 100%) downtime per period.
UDP Availability Hourly	Availability values for UDP probes. Includes Device Down and Link Down values for each 1-hour time period, expressed as a decimal value from 0.0 to 100.0 (0% to 100%) downtime per period.
TCP Availability Hourly	Availability values for TCP Connect probes. Includes Device Down and Link Down values for each 1-hour time period, expressed as a decimal value from 0.0 to 100.0 (0% to 100%) downtime per period.
DHCP Availability Hourly	Availability values for DHCP probes. Includes Device Down and Service Down values for each 1-hour time period, expressed as a decimal value from 0.0 to 100.0 (0% to 100%) downtime per period.
Hourly	Exports all hourly data. For ICMP, DNS, UDP, TCP, and DHCP both hourly latency and hourly availability are exported. For HTTP, hourly HTTP data is exported. For VoIP, hourly VoIP data is exported.

Hourly values are calculated by taking the Minimum, Maximum, and Average values for all samples that were recorded for each hour.

Availability is calculated as a percentage of time for each hour that the device or link was down.

Export Result Schema

The SlamDataExportServlet outputs the result to the browser, with the content type set to be "text/html". The output will be in XML format.

The servlet returns all of the data, from the specified start time, up to the end time. If the end time is omitted, the values are returned up to the end of the available data. Note that if an error occurs during the export process, for example a dropped database or network connection, the data output might be truncated.

The data export returns the data in a common format with standard information at the head of the data set. This information has the following components:

Results (SlcHandle, SlaHandle, (ICMPResult | UDPResult | TCPResult | DHCPResult | DNSResult | HTTPResult | VoIPResult))

ICMPLatency, VoIP, and similar terms refer to the additional DTD that is included depending on the type of data being returned. These values are described later in this chapter. If more than one device pair is found to match the query, multiple data sets may be returned.

Component	Description	Required	Quantity
SlcHandle	The numeric identifier for the requested SLC.	Yes	1
SlaHandle	The numeric identifier for the requested SLA contained within the SLC.	Yes	1

A SlmResults element has the following format:

```
<SlmResults
    slcHandle="slc_id_number"
    slaHandle="sla_id_number" >
... Specific type of data set here ...
```

The specific data set types are described in the following sections.

ICMPResult, UDPResult, DNSResult, TCPResult, DHCPResult

The format for ICMP, UDP, DNS, TCP, and DHCP Results are nearly identical. The results in each case consist of two possible data sets: latency and availability. For ICMP, UDP, and TCP Results, availability data includes device and link availability. For DNS and DHCP Results, availability data includes device and service availability. If no type was specified in the request, hourly data for both sets are returned.

ICMPResult has the following components:

ICMPResult (SourceDevice, TargetDevice, Latency, Availability)

UDPResult has the following components:

UDPResult (SourceDevice, TargetDevice, Latency, Availability)

DNSResult has the following components:

DNSResult (SourceDevice, TargetDevice, Latency, Availability)

TCPResult has the following components:

TCPResult (SourceDevice, TargetDevice, Latency, Availability)

DHCPResult has the following Components:

DHCPResult (SourceDevice, TargetDevice, Latency, Availability)

Component	Description	Required	Quantity
SourceDevice	The fully qualified hostname of the source device.	Yes	1
TargetDevice	The fully qualified hostname of the destination device.	Yes	1
Latency	Representation of one row of latency data from the database. All the data in this set is from the same time period for the given source and target devices. The Latency row format is described below.	No	0 or more
Availability	Representation of one row of availability data from the database. All the data in this set is from the same time period for the given source and target devices. The Availability row format is described below.	No	0 or more

An ICMP result might look like::

```
<ICMPResult SourceDevice="source_device_name" TargetDevice="target_device_name">
... Latency results...
... Availability results...
</ICMPResult>
```

Latency

The Latency export data format applies to general Latency data. This includes ICMP, UDP, DNS, TCP, and DHCP probes. Latency results have the following components:

Latency (Date, Min, Max, Avg, NumSuccessfulSamples, NumUnsuccessfulSamples, Status, StatusDesc)

Component	Description	MIB Values Used	Required	Quantity
Date	The stamp showing the time the data collection began. This is a date string of the form dd-MMM-yyyy:hh:mm, for example, 08-Sep-1999:15:03	N/A	Yes	1
Min	The minimum latency value, in milliseconds, for all samples in the given time period. If there is no data in that time period, this may be blank.	Minimum of rttMonhistoryCollectionSampleTime divided by the number of samples	No	0 or 1
Max	The maximum latency value, in milliseconds, for the given time period. May be the same as minimum if only one sample was recorded.	Maximum of rttMonhistoryCollectionSampleTime divided by the number of samples	No	0 or 1
Avg	The average of the latency values, in milliseconds, for the given time period. May be the same as minimum and maximum if only one sample was recorded.	Average of rttMonhistoryCollectionSampleTime divided by the number of samples	Yes	1
Num Successful Samples	The number of successful operations used to calculate Min, Max, and Avg values.	N/A	Yes	1
Num Unsuccessful Samples	The number of unsuccessful operations.	N/A	Yes	1
Status	The completion status of the SAA operations, presented as a string. See the following table for the list of possible status.	RttResponseSense	Yes	1
StatusDesc	Additional description, if available, of the completion status.	RttResponseSense	No	0 or 1

The following table shows the possible completion status and their meanings:

ok	- a valid completion occurred and timed successfully
disconnected	- the operation did not occur because the connection

```

to the target was lost
overThreshold      - a valid completion was received but the completion
time exceeded a threshold value
timeout           - an operation timed out; no completion time recorded
busy              - the operation did not occur because a previous
operation is still outstanding
notConnected       - the operation did not occur because no
connection (session) exists with the target
dropped           - the operation did not occur due to lack
of internal resource
sequenceError      - a completed operation did not contain
the correct sequence id; no completion time recorded
verifyError        - a completed operation was received, but the data it
contained did not match the expected data;
no completion time recorded
applicationSpecific
- the application generating the operation had a
specific error
dnsServerTimeout   - DNS Server Timeout
tcpConnectTimeout  - TCP Connect Timeout
httpTransactionTimeout
- HTTP Transaction Timeout
dnsQueryError      - DNS Query error (because of unknown address, etc.)
httpError          - HTTP Response StatusCode is not OK (200)
error              - Socket failures or some other errors not relevant to the
actual probe
multipleErrors      - more than one error conditions occurred

```

A Latency row might look like:

```

<Latency Date="timestamp_string" Min="15" Max="50" Avg="24"
NumSuccessfulSamples="6" NumUnsuccessfulSamples="0" status="ok" > </>

```

Availability

The Availability export data format applies to general availability data. This includes ICMP, UDP, DNS, TCP, and DHCP probes. Availability rows have the following components:

Availability (Date, DeviceDown, LinkDown)

Component	Description	MIB Values Used	Req'd	Quantity
Date	Shows the date at which data collection began. This is a string of the form dd-MMM-yyyy:hh:mm:ss ZZZ, for example, 08-Sep-1999:15:03:02 GMT	N/A	Yes	1
DeviceDown	The Source Device Down data for this row expressed as a percentage value from 0.0 (0%) to 100.0 (100%). Implicitly, if the device is down, its outgoing link is also down. However, this is not counted in the link downtime to avoid double-counting.	Lack of response is verified against sysUpTime and Reload events to distinguish between being unable to get to the device versus the device itself being down.	Yes	1
LinkDown	The Link Down data (or Service Down data in the case of DNS and DHCP) for this row, expressed as a percentage value from 0.0 (0%) to 100.0 (100%).	RttMonHistoryCollectionSense. If the value is disconnected(2), timeout(4) or notConnected(6), the link is considered to be down.	Yes	1

An Availability row has the following format:

```
<Availability Date="timestamp_string" DeviceDown="%_value_of_device_downtime"
LinkDown="%_value_of_link_downtime"> </>
```

HTTPResult

The HTTPResult contains Latency data that applies only to the special information available from HTTP probes. The HTTP data export has the following components:

HTTPResult (SourceDevice, TargetDevice, HTTPLatency)

HTTPLatency (Date, NumSuccessfulSamples, NumUnsuccessfulSamples, Status, StatusDesc, DNS, Connect, Transact, Total, ResponseSize)

DNS (min, max, avg)

Connect (min, max, avg)

Transact (min, max, avg)

Total (min, max, avg)

ResponseSize(min, max, avg)

Component	Description	MIB Values Used	Req'd	Quantity
SourceDevice	The fully qualified hostname of the source device.	N/A	Yes	1
TargetDevice	The fully qualified hostname of the destination device.	N/A	Yes	1
HTTPLatency	Representation of one row of HTTP Latency data from the database. That is, all the data in this set is from the same time period.	N/A	Yes	1 or More
Date	The stamp showing the date that data collection began.. This is a date string of the form dd-MMM-yyyy:hh:mm, for example, 08-Sep-1999:15:03	N/A	Yes	1
Num Successful Samples	The number of successful operations used to calculate Min, Max, and Avg values.	N/A	Yes	1
Num Unsuccessful Samples	The number of unsuccessful operations.	N/A	Yes	1
Status	The completion status of the SAA operations, presented as a string. See the following table for the list of possible status.	RttMonHTTPStats...	Yes	1
StatusDesc	Additional description, if available, of the completion status.	RttMonHTTPStats...	No	0 or 1
Dns	The DNS data, in milliseconds, for this row. This contains minimum,	Min, Max, and Average for time period from rttMonHTTPStatsDNSRTT	Yes	1

	maximum, and average values of the samples collected during this time period. Each of these values has an element tag.	Sum Only the Avg value is required. For hourly data, Min, Max, and Average have the same value.		
Connect	The connect time data, in milliseconds, for this row. This contains minimum, maximum, and average values of the samples collected during this time period. Each of these values has an element tag.	Min, Max, and Average for time period from rttMonHTTPStatsTCPConnectRTTSum Only the Avg value is required. For hourly data, Min, Max, and Average have the same value.	Yes	1
Transact	The HTTP transaction time data, in milliseconds, for this row. This contains minimum, maximum, and average values of the samples collected during this time period. Each of these values has an element tag.	Min, Max, and Average for time period from rttMonHTTPStatsTCPTransactionRTTSum Only the Avg value is required. For hourly data, Min, Max, and Average have the same value.	Yes	1
Total	The total time, in milliseconds, for the transaction. This is the sum of the transaction, connect, and DNS lookup times.	Min, Max, and Average for time period from rttMonHTTPStatsRTTSum Only the Avg value is required.	Yes	1
ResponseSize	The size (in octets) of the body of the HTTP response.	Min, Max, and Average for time period from rttMonHTTPStatsMessageBodyOctetSum Only the Avg value is required.	Yes	1

The following table shows the possible completion status for HTTP probes:

ok	- a valid completion occurred and timed successfully
DNSServerTimeout	- requests could not connect to the DNS Server.
TCPConnectTimeout	- requests could not connect to the HTTP Server.
TransactionTimeout	- requests timed out during HTTP transaction.
DNSQueryError	- requests had DNS Query errors.
HTTPError	- requests had HTTP errors while downloading the base page
error	- HTTP operation could not be initiated because an internal error.
busies	- HTTP operation could not be initiated because a previous HTTP operation has not been completed.
multipleErrors	- more than one error conditions occurred

An HTTP Result element will have the following format:

```
<HTTPResult SourceDevice="fully_qualified_hostname"
TargetDevice="fully_qualified_hostname" >
<HTTPLatency Date="timestamp_string" NumSuccessfulSamples="number"
    NumUnsuccessfulSamples="number" Status="status" >
  <DNS
min="min_dns_value"
max="max_dns_value"
avg="average_dns_value">
  </>
  <Connect
min="min_connect_value"
max="max_connect_value"
avg="average_connect_value" >
  </>
  <Transact
min="min_transaction_value"
max="max_transaction_value"
avg="average_transaction_value" >
  </>
  <Total
min="min_total_value"
max="max_total_value"
avg="average_total_value" >
  </>
  <ResponseSize
min="min_response_size_value"
max="max_response_size_value"
avg="average_response_size_value" >
  </>
</HTTPLatency>
</HTTPResult>
```

VoIPResult

The VoIP data export has the following components:

VoIPResult (Row)

Row (Date, NumSuccessfulSamples, NumUnsuccessfulSamples, Status, StatusDesc, FwdJitter, BwdJitter, JLatency, FwdLoss, BwdLoss)

FwdJitter (min, max, avg)

BwdJitter (min, max, avg)

JLatency (min, max, avg)

FwdLoss (min, max, avg)

BwdLoss (min, max, avg)

Component	Description	MIB Values Used	Req'd	Quantity
Row	Representation one row of VoIP data from the database. That is, all the data in this set is from the same time period.	N/A	Yes	1 or More
Date	The stamp showing the date that data collection began. This is a date string of the form dd-MMM-yyyy:hh:mm E.g., 08-Sep-1999:15:03	N/A	Yes	1
Num Successful Samples	The number of successful operations used to calculate Min, Max, and Avg values.	N/A	Yes	1
Num Unsuccessful Samples	The number of unsuccessful operations.	N/A	Yes	1
Status	The completion status of the SAA operations, presented as a string. See the following table for the list of possible status.	JitterStats...	Yes	1
StatusDesc	Additional description, if available, of the completion status.	JitterStats...	No	0 or 1
JLatency	The Latency data for this row. Min, max, and average values are reported in milliseconds.	Min, Max, and Average values for time period collected from rttMonJitterStatsRTTSum/rttMonJitterStatsNumOfRTT. For hourly data, Min, Max, and Avg have the same value.	Yes	1
FwdLoss	The Forward Packet Loss data for this row. Forward loss is collected on traffic going from the source device	Min, max, and average values from the time period collected for: RttMonJitterStatsPacketLossSD	Yes	1

	to the destination device. Min, max, and average values are reported.	$\frac{\text{rttMonJitterStatsPacketLossSD} + \text{rttMonJitterStatsNumOfRtt}}{100}$ Loss is expressed as a percentage. For hourly data, Min, Max, and Avg have the same value.		
BwdLoss	The Backward Packet Loss data for this row. Backward loss is collected on traffic going from the destination device to the source device. Min, max, and average values are reported.	Min, max, and average values from the time period collected for: $\frac{\text{RttMonJitterStatsPacketLossDS}}{\text{rttMonJitterStatsPacketLossDS} + \text{rttMonJitterStatsNumOfRtt}} \times 100$ Loss is expressed as a percentage. For hourly data, Min, Max, and Avg have the same value.	Yes	1

The following table shows the possible completion status for VoIP probes:

ok	- a valid completion occurred and timed successfully
PacketOutOfSequence	- packets arrived out of sequence.
PacketMIA	- lost packets for which we cannot determine the direction.
PacketLateArrival	- packets arrived after the timeout.
error	- operation could not be initiated because an internal error.
busies	- operation could not be initiated because a previous operation has not been completed.
multipleErrors	- more than one error conditions occurred

A VoIP element will have the following format:

```
<VoIPResult>
  <Row date="timestamp_string" NumSuccessfulSamples="number"
    NumUnsuccessfulSamples="number" status="status"  >
    <FwdJitter
      min="forward_jitter_min_value"
      max="forward_jitter_max_value"
      avg="forward_jitter_average_value" >
    </>
    <BwdJitter
      min="backward_jitter_min_value"
      max="backward_jitter_max_value"
      avg="backward_jitter_average_value" >
    </>
    <JLatency
      min="latency_min_value"
      max="latency_max_value"
      avg="latency_average_value" >
    </>
    <FwdLoss
      min="forward_min_packet_loss_value"
      max="forward_max_packet_loss_value"
      avg="forward_average_packet_loss_value" >
    </>
    <BwdLoss
      min="backward_min_packet_loss_value"
      max="backward_max_packet_loss_value"
      avg="backward_average_packet_loss_value" >
    </>
  </Row>
</VoIPResult>
```

Chapter 9 Threshold Alert Servlet

SLM configures the SAA on the source device to generate an SNMP trap if the hourly latency threshold is exceeded. Upon receipt of the trap, applications can use the Threshold Alert API to retrieve information about the SLC, SLA, and device pair associated with the trap.

Servlet API for Threshold Alert

Servlet URL

http://SLM_SERVER:PORT/CSCOnm/servlet/com.cisco.nm.slam.alert.servlet.AlertServlet

where “*SLM_SERVER*” is the hostname and “*PORT*” is the port number (default port is 1741) of the machine on which the SLM server is installed.

Note: For authentication, a valid session cookie must be included in the HTTP request to the servlet. See chapter 2.

Servlet Parameters

The table below shows the operations and parameters supported by the Threshold Alert servlet. The request's POST parameters would be structured as follows:

class=alert & cmd=Command & Parameter=Value

Threshold Alert Operations		
REQUEST		RESPONSE
Command	Parameters	Return Value (if successful)
GetAlertInfo	<p>sourceDevice=<device_name> The name or IP address of the source device that sent the SNMP trap.</p> <p>saaEntry=<SAA_entry_number> The source device SAA entry number (RTR row index) of the SAA operation that has threshold violation</p>	<p>An XML buffer that conforms to the ThresholdAlert DTD. A ThresholdAlert XML would have the following form:</p> <pre><ThresholdAlert SLAType="ICMP" > <SAAEntryNumber>123</SAAEntryNumber> <DevicePair Source="a.b.com" Target=... /> <Threshold type="HourlyLatency">100</Thresh> <SLCSLA SlcHandle="123" SlcName="SLC1" SlaHandle="234" SlaName="SLA1" /> </ThresholdAlert></pre>

For example:

<http://slm-server:1741/CSCOnm/servlet/com.cisco.nm.slam.alert.servlet.AlertServlet?class=alert&cmd=GetAlertInfo&sourceDevice=my.host.com&saaEntry=1621>

ThresholdAlert Schema

A ThresholdAlert XML has the following components:

*ThresholdAlert (SLAType, SAAEntryNumer, DevicePair, Threshold, SLCSLA)
DevicePair(Source, Target)*

Threshold(Type)

SLCSLA(SlcName, SlcHandle, SlaName, SlaHandle)

Component	Description	Required	Quantity
SLAType	The type of the SLA containing this device pair. (ICMP, UDP, DNS, HTTP, VOIP, TCP, DHCP)	Yes	1
SAAEntryNumber	The source device SAA entry number (rtr row index) of the SAA operation that has threshold violation.	No	0 or 1
DevicePair	The device pair that has the threshold violated.	Yes	1
Source	Name or IP address of the source device	Yes	1
Target	Name or IP address of the target device	Yes	1
Threshold	The type and value of the threshold that was violated	Yes	1
SLCSLA	The names and handles of the SLC and SLA that contain the device pair. A device pair can be shared by more than one SLC/SLA.	Yes	1 or more

Here is an example of an XML buffer that could be returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ThresholdAlert SYSTEM "http://slam:1741/slam/DTD/ThresholdAlert.dtd">
<ThresholdAlert SLAType="ICMP">
  <SAAEntryNumber>1</SAAEntryNumber>
  <DevicePair Source="192.168.0.1" Target="192.168.0.2"/>
  <Threshold Type="HourlyLatency">200</Threshold>
  <SLCSLA SlcHandle="23" SlcName="myslc" SlaHandle="33" SlaName="mysla"/>
  <SLCSLA SlcHandle="39" SlcName="slc2" SlaHandle="49" SlaName="sla2"/>
</ThresholdAlert>
```

By looking at this XML buffer you learn the following:

1. The SLA type that was affected.
2. SAA Entry number.
3. The Source and target device.
4. The threshold that was violated.
5. The SLC and SLAs that were affected.

Trap troubleshooting:

If you know you are passing the correct information but you are getting nothing back, or you are getting incorrect information, one of the following could be the cause:

- There has been a change in the SAA index associated with this SLA – the job could have disappeared from the source router because of an RTR reset or a router reload and the job will then be recreated with another SAA index. Because you could looking at the wrong SLA. so always be careful because this is dynamic data.
- The probe might no longer be working. Perhaps this trap happened as the network was crashing and now the probe isn't working, resulting in the SLAM server no longer having an SAA index.

This is an example of what a received trap will look like:

```
Jan 22 22:03:25 slam snmptrapd[2942]: 192.168.0.1: Enterprise Specific Trap (3) Uptime: 11
days, 0:40:14.36, enterprises.9.9.42.1.2.1.1.3.18 = "18" Hex: 31 38 ,
enterprises.9.9.42.1.4.1.1.5.18.1.1384.1 = "", enterprises.9.9.42.1.2.9.1.7.18.1.1384.1 = 2
```

The following is a description of the OIDs returned in a trap:

OID	Description
.1.3.6.1.4.1.9.9.42.1.2.1.1.3 = rttMonCtrlAdminTag (aka Rtr/SAA Row Index)	A string that is used by a managing application to identify the RTT target. This string is inserted into trap notifications but has no other significance to the agent.
.1.3.6.1.4.1.9.9.42.1.4.1.1.5 = rttMonHistoryCollectionAddress	When the RttMonRttType is 'echo' or 'pathEcho', this string specifies the address of the target for this RTT operation. For all other values of RttMonRttType, this string is null.
.1.3.6.1.4.1.9.9.42.1.2.9.1.7 = rttMonCtrlOperOverThresholdOccurred	This object changes its value for all RttMonRttTypes. This object is changed by operation completion times over threshold, as defined by rttMonReactAdminThresholdType. When this value changes, a reaction could occur, as defined by rttMonReactAdminThresholdType. If a trap is sent. it is a rttMonThresholdNotification.

Chapter 10 Report Navigator Cross-Launch

It is possible to launch the SLM Report Navigator from outside the SLM application. First you must set up an authentication session, as described in Chapter 2.. After the session is established, launch the Navigator by calling the following URL:

http://SLM_SERVER:PORT/slam/jsp/reportNav.jsp

where "SLM_SERVER" is the hostname and "PORT" is the port number (default port number is 1741) of the machine on which the SLM server is installed.

Chapter 11 View Collection Manager Status

Use the following URL to view the status of the distributed CMs that collect data for the SLM server. First you must set up an authentication session, as described in Chapter 2. After the session is established, launch the Status page by calling the following URL:

http://SLM_SERVER:PORT/slam/jsp/meStatus.jsp

where "SLM_SERVER" is the hostname and "PORT" is the port number (default port number is 1741) of the machine on which the SLM server is installed.

Chapter 12 Putting It All Together

In this chapter we will assemble the services described in the previous chapters into one comprehensive example.

Using Cookies

The code below shows how a Java application might communicate with the SLM server:

```
/* Copyright (c) 1999-2000 Cisco Systems, Inc. All rights reserved */

import java.io.*;
import java.net.*;

/**
 * Class to access SLM Folder/SLC through HTTP Post w/ session cookies.
 * Does the following:
 * + Login to CMF Authentication Servlet from a stand-alone app
 * + Get the session cookie
 * + Use the session cookie in the HTTP requests to Slam servlet
 * + Logout
 */

public class slam
{
    protected String _urlString, _logout, _login;
    protected static final int BUF_SIZE = 2000;

    /* Constructor */
    public slam(String hostURL)
    {
        _urlString = hostURL +
        "/CSCOnm/servlet/com.cisco.nm.slam.admin.servlet.AdminServlet";
        _login = hostURL +
        "/CSCOnm/servlet/com.cisco.nm.cmf.servlet.CsAuthServlet?cmd=authUser&name=admin&pwd=YWRtaW4=";
        _logout = hostURL +
        "/CSCOnm/servlet/com.cisco.nm.cmf.servlet.CsAuthServlet?cmd=logout";
        System.out.println("Servlet URL: " + _urlString);
    }

    /* Do a HTTP GET request */
    public String doGet(String urlString, String sessionID)
    {
        DataInputStream in = null;
        try
        {
            URL url = new URL(urlString);
            URLConnection conn = url.openConnection();
            conn.setUseCaches(false);
            conn.setDefaultUseCaches(false);
            conn.setRequestProperty("Content-Type",
            "application/x-www-form-urlencoded");
            if(sessionID !=null)
            conn.setRequestProperty("COOKIE", "jrunsessionid=" + sessionID);
        }
    }
}
```

```

in = new DataInputStream(new BufferedInputStream(conn.getInputStream()));

String res = in.readLine();
in.close();
return res;
}
catch (Exception excp)
{
    excp.printStackTrace();
return null;
}
}

/* Do a HTTP POST request */
public String doPost(String sessionID, String clas, String cmd, String param, String
data)
{
    String      result;
    StringBuffer buf;
    BufferedReader in = null;
    PrintWriter  out= null;

    try
    {
        URL url = new URL(_urlString);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("POST");
        conn.setDoOutput(true);
        conn.setDoInput(true);
        conn.setUseCaches(false);
        conn.setDefaultUseCaches(false);
        conn.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
        conn.setRequestProperty("COOKIE", "jrunsessionid=" + sessionID);

        // send request's body
        out = new PrintWriter(conn.getOutputStream(), true);
        String req = "class=" + clas + "&" + "cmd=" + cmd;
        if(param!=null)
            req += "&" + param;
        if(data!=null)
            req += "&data=" + URLEncoder.encode(data);
        out.print(req);
        out.flush();
        out.close();
        System.out.println("POST Request sent: " + req);

        // receive response (XML)
        in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
        buf = new StringBuffer(BUF_SIZE);
        while((data = in.readLine()) != null)
            buf.append(data + "\n");
        result = buf.toString();
        in.close();
    }
}

```

```

return result;
}
catch (Exception excp)
{   excp.printStackTrace();
try
{   out.close();
in.close();
}
catch (Exception ex)
{   ex.printStackTrace();
}
return null;
}
}

/* main */
public static void main(String argv[])
{
String cmd=null, clas=null, param=null, data=null, line;
StringBuffer buff = new StringBuffer(BUF_SIZE);

if(argv.length < 3)
{   System.out.println("Usage: java Slam <slamserver> Folder|SLC <cmd> [param]
[xmlfile]");
return;
}
cmd = argv[2];
clas = argv[1];
if(argv.length >=4)
param = argv[3];
if(argv.length >=5)
{   try
{   BufferedReader f = new BufferedReader(new FileReader(argv[4]));
while ((line = f.readLine())!=null)
{   buff.append(line+"\n");
}
}
data = buff.toString();
}
catch(Exception ex)
{   ex.printStackTrace();
data = null;
}
}

// Perform HTTP request
try
{
slam sa = new slam("http://"+argv[0]);

java.lang.Thread.sleep(3000);
String res = sa.doGet(sa._login, null);    // login
System.out.println("***Login: " + res);

int idx = res.lastIndexOf(':');    // get session ID

```

```

String session = res.substring(idx+1);
System.out.println("****Session ID: " + session);

java.lang.Thread.sleep(7000);           // perform request(s) w/ session ID
String result = sa.doPost(session, clas, cmd, param, data);
System.out.println("****Response1:\n" + result);

java.lang.Thread.sleep(7000);           // this request will fail w/out a valid session ID
result = sa.doPost("badsession", clas, cmd, param, data);
System.out.println("****Response2:\n" + result);

java.lang.Thread.sleep(7000);
sa.doGet(sa._logout, session);           // logout
System.out.println("****Logout - Bye !");
}
catch (Exception ex)
{   ex.printStackTrace();
}
}
}

```

For simplicity, all the following examples are shown only as the Post command that is required for the request. If you were performing these requests in conjunction with the Java sample shown above, you would embed the requests into the command strings of the code. Note that not all commands go to the same servlet, so you might need to specify a different servlet than the one shown in the example code.

Getting Device Lists

Devices must be in the Essentials Inventory database before SLM can monitor them. Also, the capabilities of the device must match the types of metrics that are to be collected. You can use the Admin servlet to retrieve a list of available devices prior to defining an SLC. You can further specify the type of device metrics you are interested in to ensure the devices are capable of handling the desired metric type.

Device List Request

The following request finds devices that are managed by the Essentials Inventory and are capable of returning ICMP Metrics.

```

POST http://myhost:1741/CSCOnm/servlet/com.cisco.nm.slam.admin.servlet.AdminServlet
HTTP/1.0

Accept: text/html
. . .
Content-type: application/x-www-form-urlencoded

class=Inventory & cmd=GetDevices & Type=ICMP

```

Device List Response

A list of fully qualified device names are returned in XML format. The hostname is returned if it is available. Otherwise, the IP Address is returned.

```
<?xml version="1.0"?>
<!DOCTYPE DeviceList SYSTEM DeviceList.dtd>
<DeviceList type=ICMP >
<Device> "myhostname.mydomain.com" </ >
<Device> "anotherhostname.mydomain.com" </ >
<Device> "123.11.12.13" </>
</DeviceList>
```

Defining SLCs and SLAs

Request to Add a New SLC

The following command adds a new SLC to the folder whose handle number is 9876. If the folder ID is omitted, the SLC is created in the default folder.

```
POST http://myhost:1741/CSCOnm/servlet/com.cisco.nm.slam.admin.servlet.AdminServlet
HTTP/1.0

Accept: text/html
. . .
Content-type: application/x-www-form-urlencoded

class=Slc & cmd=Add & folder=9876 & data="...URL-encoded XML String of the SLC
definition you want to create..."
```

Response to Add Request

If successful, the SLC handle number is returned. This handle can be used to further manipulate the SLC definition or to retrieve the resulting SLC data. Note that it could take several hours for the data to be collected and made available on the SLM server.

```
<SlamAdminResult>
1234
</SlamAdminResult>
```

Getting Existing SLC Handles and Names

The SLC handles are returned when you define a new SLC. Alternatively, you can list all SLCs by getting an SLC or Folder List, find the SLC you want by name, and the associated handle returned in the list.

Using SLC Lists

SLC List Request

```
POST http://myhost:1741/CSCOnm/servlet/com.cisco.nm.slam.admin.servlet.AdminServlet
HTTP/1.0
Accept: text/html
. . .
Content-type: application/x-www-form-urlencoded
class=Slc & cmd=Enumerate
```

SLC List Response:

```

<SlcList>
<Slc handle="7654">
<Name>"SLC Number 1"</Name>
<LastModifiedTime> "08-Sep-1999:15:03" </LastModifiedTime>
</Slc>
<Slc handle="8765">
<Name>"Another SLC"</Name>
<LastModifiedTime> "08-Sep-1999:15:03" </LastModifiedTime>
</Slc>
<Slc handle="9876">
<Name>"Yet Another SLC"</Name>
<LastModifiedTime> "08-Sep-1999:15:34" </LastModifiedTime>
</Slc>
</SlcList>

```

Using Folder Lists

If you are interested only in the SLCs of a particular folder, you can use the Folder List rather than the SLC List mechanism. If you omit the handle from the Folder List request, all folders and their associated SLCs are returned.

Folder List Request

```

POST http://myhost:1741/CSCOnm/servlet/com.cisco.nm.slam.admin.servlet.AdminServlet
HTTP/1.0

Accept: text/html
. . .
Content-type: application/x-www-form-urlencoded

class=Folder & cmd=Get & folderhandle="1234"

```

Folder List Response

```

<FolderList>
<Folder handle="1234">
<Name>"A Folder Name"</Name>
<SlcList>
<Slc handle="8765">
<Name>"Another SLC"</Name>
<LastModifiedTime> "08-Sep-1999:15:03" </LastModifiedTime>
</Slc>
<Slc handle="9876">
<Name>"Yet Another SLC"</Name>
<LastModifiedTime> "08-Sep-1999:15:34" </LastModifiedTime>
</Slc>
</SlcList>
</Folder>
</FolderList>

```

Getting SLA Handles and Names from an Existing SLC

To get information about the SLAs included in an SLC, you must retrieve the SLC definition by using its handle. This will return the complete SLC description in XML format and will include all the SLA handles, names, and definitions for that SLC.

SLC Request

```
POST http://hostname:1741/CSCOnm/servlet/com.cisco.nm.slam.admin.servlet.
AdminServlet HTTP/1.0
Accept: text/html
. . .
Content-type: application/x-www-form-urlencoded

class=Slc & cmd=Get & slchandle=1234
```

SLC Response

```
<Slc Enabled="true" Handle="1234">
<Name>ISP Service SLC 4</Name>
<Comment>Testing SLC</Comment>
<ApplyTime Zone="LOCAL">
<ApplyMon>
<FromTime>8</FromTime>
<ToTime>17</ToTime>
</ApplyMon>
<ApplyTue>
<FromTime>8</FromTime>
<ToTime>17</ToTime>
</ApplyTue>
<ApplyWed>
<FromTime>8</FromTime>
<ToTime>17</ToTime>
</ApplyWed>
<ApplyThu>
<FromTime>8</FromTime>
<ToTime>17</ToTime>
</ApplyThu>
<ApplyFri>
<FromTime>8</FromTime>
<ToTime>17</ToTime>
</ApplyFri>
<ApplySat>
<FromTime>0</FromTime>
<ToTime>0</ToTime>
</ApplySat>
<ApplySun>
<FromTime>0</FromTime>
<ToTime>0</ToTime>
</ApplySun>
</ApplyTime>
<Sla Handle="9876">
<Name>SLA9</Name>
<Comment>This is a demo Round Trip Report SLA</Comment>
<CreateTime>Tue Aug 17 13:10:40 PDT 1999</CreateTime>
<ICMPMetric SamplingInterval="1" TOS="1">
```

```

<ICMPThreshold>
<AvgHourlyLatency>10</AvgHourlyLatency>
<AvgDailyLatency>10</AvgDailyLatency>
<DailyAvailability>70</DailyAvailability>
<MonthlyAvailability>70</MonthlyAvailability>
</ICMPThreshold>
<ICMPDeviceSpec>
<SourceDevice> host9.cisco.com </SourceDevice>
<TargetDevice> host9.ibm.net</TargetDevice>
<CreateTime> 10-25-1999 14:06:07 GMT </CreateTime>
</ICMPDeviceSpec>
<ICMPDeviceSpec>
<SourceDevice> host8.cisco.com </SourceDevice>
<TargetDevice>host8.ibm.net</TargetDevice>
<CreateTime> 10-25-1999 14:06:20 GMT </CreateTime>
</ICMPDeviceSpec>
</ICMPMetric>
</Sla>
<LastModifiedTime>Sat Aug 12 01:23:00 PDT 1995</LastModifiedTime>
</Slc>

```

Retrieving SLC/SLA Data

When you have the SLC and SLA handles, you can use this information to retrieve the available data via the SlamDataExportServlet. You can either retrieve information about all devices included in an SLA or you can specify a particular device pair in that SLA. You can also specify time ranges to limit the amount of data that is returned.

SLA Data Request

```

POST http://hostname:1741/CSCOnm/servlet/com.cisco.nm.slam.report.servlet.
SlamDataExportServlet HTTP/1.0
Accept: text/html
. . .
Content-type: application/x-www-form-urlencoded

slcid=4 & slaid=24 & srcdev=name1.cisco.com & destdev=name2.cisco.com & start="08-Sep-
1999:15:00" & end="08-Sep-1999:17:00"

```

SLA Data Response

Because no Type was specified in the request, both the Latency and Availability hourly data are returned for the requested time period:

```
<Results slcHandle="12" slaHandle="3" >
  <ICMPResult SourceDevice="MyHost.com" TargetDevice="YourHost.com">
    <Latency Date="08-Sep-1999:15:00" Min="2" Max="23" Avg="14"
      NumSuccessfulSamples=60 NumUnsuccessfulSamples=0 Status="ok"> </>
    <Latency Date="08-Sep-1999:16:00" Min="7" Max="44" Avg="26"
      NumSuccessfulSamples=48 NumUnsuccessfulSamples=12 Status="ok"> </>
    <Latency Date="08-Sep-1999:17:01" Min="5" Max="15" Avg="11"
      NumSuccessfulSamples=60 NumUnsuccessfulSamples=0 Status="ok"> </>
    <Availability Date="08-Sep-1999:15:01" Device="0.0" Link="0.0"> </>
    <Availability Date="08-Sep-1999:16:00" Device="0.2" Link="0.0"> </>
    <Availability Date="08-Sep-1999:17:02" Device="0.0" Link="0.0"> </>
  </ICMPResult>
  <ICMPResult SourceDevice="AnotherHost.com" TargetDevice="YourHost.com">
    <Latency Date="08-Sep-1999:15:00" Min="1" Max="13" Avg="4"
      NumSuccessfulSamples=60 NumUnsuccessfulSamples=0 Status="ok"> </>
    <Latency Date="08-Sep-1999:16:00" Min="3" Max="24" Avg="16"
      NumSuccessfulSamples=48 NumUnsuccessfulSamples=12 Status="ok"> </>
    <Latency Date="08-Sep-1999:17:01" Min="2" Max="17" Avg="9"
      NumSuccessfulSamples=60 NumUnsuccessfulSamples=0 Status="ok"> </>
    <Availability Date="08-Sep-1999:15:01" Device="0.0" Link="0.0"> </>
    <Availability Date="08-Sep-1999:16:00" Device="0.2" Link="0.0"> </>
    <Availability Date="08-Sep-1999:17:02" Device="0.0" Link="0.0"> </>
  </ICMPResult>
</Results>
```

Appendix A XML Document Type Definitions (DTDs)

The following DTDs can be viewed from a SLM server via the URL:

http://SLM_SERVER:PORT/slam/DTD/fileName.dtd

where "*SLM_SERVER*" is the hostname and "*PORT*" is the port number (if applicable) of the machine on which the SLM server is installed.

For example, to view the FolderList file via the SLM server on port 1741 of the host called MyServerName, you would enter:

<http://MyServerName:1741/slam/DTD/FolderList.dtd>

FolderList.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 2000-2001 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->
<!-- Revision: 2.0 FolderList.dtd -->

<!ELEMENT FolderList (Folder*)>

<!ELEMENT Folder (Name, SlcList)>
<!ATTLIST Folder Handle CDATA #IMPLIED>

<!ELEMENT SlcList (Slc*)>

<!ELEMENT Slc (Name, LastModifiedTime)>
<!ATTLIST Slc Handle CDATA #IMPLIED>

<!ELEMENT Name (#PCDATA)>
<!ELEMENT LastModifiedTime (#PCDATA)>
<!-- LastModifiedTime format: DD-MMM-YYYY hh:mm:ss GMT -->
<!-- For example: 20-Dec-1999 08:51:49 GMT -->
```

SLCList.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 2000-2001 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!-- Revision: 2.0 SlcList.dtd -->

<!ELEMENT SlcList (Slc*)>

<!ELEMENT Slc (Name, LastModifiedTime)>
<!ATTLIST Slc Handle CDATA #IMPLIED>

<!ELEMENT Name (#PCDATA)>
<!ELEMENT LastModifiedTime (#PCDATA)>
<!-- LastModifiedTime format: DD-MMM-YYYY hh:mm:ss GMT -->
<!-- For example: 20-Dec-1999 08:51:49 GMT -->
```

SLC.dtd

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 1999,2000 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ENTITY % UDPMetricEntity SYSTEM "UDPMetric.dtd">
<!ENTITY % ICMPMetricEntity SYSTEM "ICMPMetric.dtd">
<!ENTITY % HTTPMetricEntity SYSTEM "HTTPMetric.dtd">
<!ENTITY % DNSMetricEntity SYSTEM "DNSMetric.dtd">
<!ENTITY % VoIPMetricEntity SYSTEM "VoIPMetric.dtd">
<!ENTITY % TCPMetricEntity SYSTEM "TCPMetric.dtd">
<!ENTITY % DHCPMetricEntity SYSTEM "DHCPMetric.dtd">

<!-- Slc -->
<!ELEMENT Slc (Name, Comment?, Sla+, ApplyTime, LastModifiedTime?)>
<!ATTLIST Slc Handle CDATA #IMPLIED>
<!ATTLIST Slc Enabled (true | false) "true">
<!ATTLIST Slc TimeZone CDATA #IMPLIED>
<!-- The time zone based on which daily rollup/reports are done. -->
<!-- Currently only the server local time zone is supported. -->

<!ELEMENT Name (#PCDATA)>
<!ELEMENT Comment (#PCDATA)>

<!-- ApplyTime for Slc -->
<!-- If FromTime and ToTime are both 0, it means that Slc -->
<!-- is not applied for that day. If FromTime is 0 and -->
<!-- ToTime is 24, then the Slc is applied all day. -->
<!-- If Zone is LOCAL, the server local time will be used -->
<!-- to determined the apply time. If Zone is UTC, GMT +0 -->
<!-- will be used to determined the apply time. -->
<!-- Current version supports only LOCAL Zone. -->
<!ELEMENT ApplyTime (ApplyMon, ApplyTue, ApplyWed, ApplyThu, ApplyFri, ApplySat,
ApplySun)>
<!ATTLIST ApplyTime Zone (LOCAL) #REQUIRED>
<!ELEMENT ApplyMon (FromTime, ToTime)>
<!ELEMENT ApplyTue (FromTime, ToTime)>
<!ELEMENT ApplyWed (FromTime, ToTime)>
<!ELEMENT ApplyThu (FromTime, ToTime)>
<!ELEMENT ApplyFri (FromTime, ToTime)>
<!ELEMENT ApplySat (FromTime, ToTime)>
<!ELEMENT ApplySun (FromTime, ToTime)>

<!ELEMENT FromTime (#PCDATA)>
<!ELEMENT ToTime (#PCDATA)>

<!ELEMENT LastModifiedTime (#PCDATA)>
<!-- LastModified/CreateTime format is DD-MMM-YYYY hh:mm:ss GMT -->
<!-- For example: 20-Dec-1999 08:51:49 GMT -->

<!-- Tags used for Device Specifications -->
<!ELEMENT SourceDevice (#PCDATA)>

```

```

<!-- SourceDevice WriteCommStr CData #IMPLIED -->
<!-- TargetDevice (#PCDATA) -->
<!-- CreateTime (#PCDATA) -->

<!-- Sla -->
<!-- (Name, Comment?, LastModifiedTime?, (UDPMetric | ICMPMetric | HTTPMetric
| DNSMetric | VoIPMetric | DHCPMetric | TCPMetric)) -->
<!-- Sla Handle CData #IMPLIED -->

%UDPMetricEntity;
%ICMPMetricEntity;
%DNSMetricEntity;
%HTTPMetricEntity;
%VoIPMetricEntity;
%TCPMetricEntity;
%DHCPMetricEntity;

<!-- Tags used for thresholds -->
<!-- AvgHourlyLatency (#PCDATA) -->
<!-- AvgHourlyLatency unit CData #FIXED "msec" -->
<!-- AvgDailyLatency (#PCDATA) -->
<!-- AvgDailyLatency unit CData #FIXED "msec" -->
<!-- DailyAvailability (#PCDATA) -->
<!-- MonthlyAvailability (#PCDATA) -->

```

UDPMetric.dtd

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- (C) Copyright 1999 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->
<!-- DTD for UDP metric -->

<!-- UDPMetric (UDPThreshold, UDPDeviceSpec+) -->
<!-- UDPMetric SamplingInterval (1|5|10|15|30) "5" -->
<!-- UDPMetric PayloadSize CData #IMPLIED -->
<!-- UDPMetric Port CData #IMPLIED -->
<!-- UDPMetric TOS CData #IMPLIED -->

<!-- UDPDeviceSpec (SourceDevice, TargetDevice, CreateTime?) -->

<!-- UDPThreshold
(AvgHourlyLatency,AvgDailyLatency,DailyAvailability,MonthlyAvailability) -->

```

ICMPMetric.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- (C) Copyright 1999 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->
<!-- DTD for ICMP metric -->

<!ELEMENT ICMPMetric (ICMPThreshold, ICMPDeviceSpec+)>
<!ATTLIST ICMPMetric SamplingInterval (1|5|10|15|30) "5">
<!ATTLIST ICMPMetric PayloadSize CDATA #IMPLIED>
<!ATTLIST ICMPMetric TOS CDATA #IMPLIED>

<!ELEMENT ICMPDeviceSpec (SourceDevice, TargetDevice, CreateTime?)>

<!ELEMENT ICMPThreshold
(AvgHourlyLatency,AvgDailyLatency,DailyAvailability,MonthlyAvailability)>
```

DNSMetric.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- (C) Copyright 1999 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->
<!-- DTD for DNS metric -->

<!ELEMENT DNSMetric (TestIPAddr, DNSThreshold, DNSDeviceSpec+)>
<!ATTLIST DNSMetric SamplingInterval (1|5|10|15|30) "5" >
<!ELEMENT TestIPAddr (#PCDATA)>

<!ELEMENT DNSDeviceSpec (SourceDevice, TargetDevice, CreateTime?)>

<!ELEMENT DNSThreshold
(AvgHourlyLatency,AvgDailyLatency,DailyAvailability,MonthlyAvailability)>
```

HTTPMetric.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- (C) Copyright 1999-2000 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->
<!-- DTD for HTTP metric -->

<!ELEMENT HTTPMetric (URLPath?, ProxyServer?, NameServer?, HTTPThreshold,
HTTPDeviceSpec+)>
<!-- ATTLIST HTTPMetric Port CDATA #IMPLIED>
<!-- ATTLIST HTTPMetric SamplingInterval (1|5|10|15|30) "5">
<!-- ATTLIST HTTPMetric CacheEnable (true | false) "false">
<!-- ATTLIST HTTPMetric TOS CDATA #IMPLIED >
<!-- Timeout in msec; default is 5000 -->
<!-- ATTLIST HTTPMetric Timeout CDATA #IMPLIED >
<!-- ELEMENT URLPath (#PCDATA)>
<!-- ELEMENT ProxyServer (#PCDATA)>
<!-- ELEMENT NameServer (#PCDATA)>

<!-- ELEMENT HTTPDeviceSpec (SourceDevice, TargetDevice, CreateTime?)>

<!-- ELEMENT HTTPThreshold (AvgHourlyLatency,AvgDailyLatency)>
```

VoIPMetric.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- (C) Copyright 1999 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->
<!-- DTD for voice-over-IP metric -->

<!-- ELEMENT VoIPMetric (VoIPThreshold, VoIPDeviceSpec+)>
<!-- ATTLIST VoIPMetric SamplingInterval (1|5|10|15|30) "5" >
<!-- ATTLIST VoIPMetric PacketsPerSample CDATA #REQUIRED >
<!-- ATTLIST VoIPMetric InterPacketInterval CDATA #REQUIRED >
<!-- ATTLIST VoIPMetric PayloadSize CDATA #IMPLIED >
<!-- ATTLIST VoIPMetric EnableControl (true | false) "true" >
<!-- ATTLIST VoIPMetric TOS CDATA #IMPLIED >
<!-- ATTLIST VoIPMetric SourcePort CDATA #REQUIRED>
<!-- ATTLIST VoIPMetric TargetPort CDATA #REQUIRED>
<!-- Note: The unit for SamplingInterval is minute -->
<!-- The unit for InterPacketInterval is milisecond -->

<!-- ELEMENT VoIPDeviceSpec (SourceDevice, TargetDevice, CreateTime?)>

<!-- ELEMENT VoIPThreshold (Jitter, RoundTripLatency, PacketLoss) >
<!-- ELEMENT Jitter (#PCDATA) >
<!-- ELEMENT RoundTripLatency (#PCDATA) >
<!-- ELEMENT PacketLoss (#PCDATA) >
```

TCPMetric.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- (C) Copyright 2000 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->
<!-- DTD for TCP metric -->

<!ELEMENT TCPMetric (TCPThreshold, TCPDeviceSpec+)>
<!ATTLIST TCPMetric SamplingInterval (1|5|10|15|30) "5">
<!ATTLIST TCPMetric Port CDATA #IMPLIED>
<!ATTLIST TCPMetric ResponderEnable (true | false) "false">
<!ATTLIST TCPMetric TOS CDATA #IMPLIED >

<!ELEMENT TCPDeviceSpec (SourceDevice, TargetDevice, CreateTime?)>

<!ELEMENT TCPThreshold
(AvgHourlyLatency,AvgDailyLatency,DailyAvailability,MonthlyAvailability)>
```

DHCPMetric.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- (C) Copyright 2000 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->
<!-- DTD for DHCP metric -->

<!ELEMENT DHCPMetric (DHCPThreshold, DHCPDeviceSpec+)>
<!ATTLIST DHCPMetric SamplingInterval (1|5|10|15|30) "5">
<!ATTLIST DHCPMetric Broadcast (true) "true">
<!-- Timeout in msec; default is 1000 -->
<!ATTLIST DHCPMetric Timeout CDATA #IMPLIED>

<!ELEMENT DHCPDeviceSpec (SourceDevice, TargetDevice, CreateTime?)>
<!ELEMENT DHCPThreshold
(AvgHourlyLatency,AvgDailyLatency,DailyAvailability,MonthlyAvailability)>
```

SlamError.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- (C) Copyright 1999 Cisco Systems, Inc. -->
<!-- All Rights Reserved -->
<!-- DTD for Slam Error -->

<!ELEMENT SlamError (Descr, Cause?, Action?) >
<!ATTLIST SlamError Code CDATA #REQUIRED>
<!ATTLIST SlamError Sev CDATA #REQUIRED>
<!ELEMENT Descr (#PCDATA) >
<!ELEMENT Cause (#PCDATA) >
<!ELEMENT Action (#PCDATA) >
```

SlamAdminResult.dtd

```
<?xml encoding="UTF-8" version="1.0"?>
<!-- (C) Copyright 2000-2001 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ELEMENT SlamAdminResult #PCDATA>
```

SlmResults.dtd:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 1999 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ENTITY % ICMPResultEntity SYSTEM "ICMPResult.dtd">
<!ENTITY % UDPResultEntity SYSTEM "UDPResult.dtd">
<!ENTITY % TCPResultEntity SYSTEM "TCPResult.dtd">
<!ENTITY % DHCPResultEntity SYSTEM "DHCPResult.dtd">
<!ENTITY % DNSResultEntity SYSTEM "DNSResult.dtd">
<!ENTITY % HTTPResultEntity SYSTEM "HTTPResult.dtd">
<!ENTITY % VoIPResultEntity SYSTEM "VoIPResult.dtd">

%ICMPResultEntity;
%UDPResultEntity;
%TCPResultEntity;
%DHCPResultEntity;
%DNSResultEntity;
%HTTPResultEntity;
%VoIPResultEntity;

<!-- SlmResults -->
<!ELEMENT SlmResults ((ICMPResult* | UDPResult* | DHCPResult* | TCPResult* | DNSResult*
| HTTPResult* | VoIPResult*))>
<!ATTLIST SlmResults SlcHandle CDATA #REQUIRED>
```

```

<!ATTLIST SlmResults SlaHandle CDATA #REQUIRED>

<!-- Availability subset -->
<!ELEMENT Availability EMPTY>
<!ATTLIST Availability Date CDATA #REQUIRED>
<!ATTLIST Availability DeviceDown CDATA #REQUIRED>
<!ATTLIST Availability LinkDown CDATA #REQUIRED>

<!-- Generic Latency subset -->
<!ELEMENT Latency EMPTY>
<!ATTLIST Latency Date CDATA #REQUIRED>
<!ATTLIST Latency Min CDATA #IMPLIED>
<!ATTLIST Latency Max CDATA #IMPLIED>
<!ATTLIST Latency Avg CDATA #REQUIRED>
<!ATTLIST Latency NumSuccessfulSamples CDATA #REQUIRED>
<!ATTLIST Latency NumUnsuccessfulSamples CDATA #REQUIRED>
<!ATTLIST Latency Status CDATA #REQUIRED>
<!ATTLIST Latency StatusDesc CDATA #IMPLIED>

```

ICMPResult.dtd:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 1999 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ELEMENT ICMPResult (Latency*, Availability*)>
<!ATTLIST ICMPResult SourceDevice CDATA #REQUIRED>
<!ATTLIST ICMPResult TargetDevice CDATA #REQUIRED>

```

UDPResult.dtd:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 1999 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ELEMENT UDPResult (Latency*, Availability*)>
<!ATTLIST UDPResult SourceDevice CDATA #REQUIRED>
<!ATTLIST UDPResult TargetDevice CDATA #REQUIRED>

```

DNSResult.dtd:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 1999 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ELEMENT DNSResult (Latency*, Availability*)>
<!ATTLIST DNSResult SourceDevice CDATA #REQUIRED>
<!ATTLIST DNSResult TargetDevice CDATA #REQUIRED>

```

TCPResult.dtd:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 2000 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ELEMENT TCPResult (Latency*, Availability*)>
<!ATTLIST TCPResult SourceDevice CDATA #REQUIRED>
<!ATTLIST TCPResult TargetDevice CDATA #REQUIRED>
```

DHCPResult.dtd:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 2000 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ELEMENT DHCPResult (Latency*, Availability*)>
<!ATTLIST DHCPResult SourceDevice CDATA #REQUIRED>
<!ATTLIST DHCPResult TargetDevice CDATA #REQUIRED>
```

HTTPResult.dtd:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 1999,2000 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ELEMENT HTTPResult (HTTPLatency*)>
<!ATTLIST HTTPResult SourceDevice CDATA #REQUIRED>
<!ATTLIST HTTPResult TargetDevice CDATA #REQUIRED>

<!ELEMENT HTTPLatency (Total, DNS, Connect, Transact, HTTPResponseSize)>
<!ATTLIST HTTPLatency Date CDATA #REQUIRED>
<!ATTLIST HTTPLatency NumSuccessfulSamples CDATA #REQUIRED>
<!ATTLIST HTTPLatency NumUnsuccessfulSamples CDATA #REQUIRED>
<!ATTLIST HTTPLatency Status CDATA #REQUIRED>
<!ATTLIST HTTPLatency StatusDesc CDATA #IMPLIED>

<!ELEMENT Total EMPTY>
<!ATTLIST Total Min CDATA #IMPLIED>
<!ATTLIST Total Max CDATA #IMPLIED>
<!ATTLIST Total Avg CDATA #REQUIRED>

<!ELEMENT DNS EMPTY>
<!ATTLIST DNS Min CDATA #IMPLIED>
<!ATTLIST DNS Max CDATA #IMPLIED>
<!ATTLIST DNS Avg CDATA #REQUIRED>

<!ELEMENT Connect EMPTY>
<!ATTLIST Connect Min CDATA #IMPLIED>
<!ATTLIST Connect Max CDATA #IMPLIED>
<!ATTLIST Connect Avg CDATA #REQUIRED>

<!ELEMENT Transact EMPTY>
```

```

<!ATTLIST Transact Min CDATA #IMPLIED>
<!ATTLIST Transact Max CDATA #IMPLIED>
<!ATTLIST Transact Avg CDATA #REQUIRED>

<!ELEMENT HTTPResponseSize EMPTY>
<!ATTLIST HTTPResponseSize Min CDATA #IMPLIED>
<!ATTLIST HTTPResponseSize Max CDATA #IMPLIED>
<!ATTLIST HTTPResponseSize Avg CDATA #REQUIRED>

```

VoIPResult.dtd:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 1999, 2000 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ELEMENT VoIPResult (Row*)>
<!ATTLIST VoIPResult SourceDevice CDATA #REQUIRED>
<!ATTLIST VoIPResult TargetDevice CDATA #REQUIRED>

<!ELEMENT Row (FwdJitter, BwdJitter, JLatency, FwdLoss, BwdLoss)>
<!ATTLIST Row Date CDATA #REQUIRED>
<!ATTLIST Row NumSuccessfulSamples CDATA #REQUIRED>
<!ATTLIST Row NumUnsuccessfulSamples CDATA #REQUIRED>
<!ATTLIST Row Status CDATA #REQUIRED>
<!ATTLIST Row StatusDesc CDATA #IMPLIED>

<!ELEMENT FwdJitter EMPTY>
<!ATTLIST FwdJitter Min CDATA #IMPLIED>
<!ATTLIST FwdJitter Max CDATA #IMPLIED>
<!ATTLIST FwdJitter Avg CDATA #REQUIRED>

<!ELEMENT BwdJitter EMPTY>
<!ATTLIST BwdJitter Min CDATA #IMPLIED>
<!ATTLIST BwdJitter Max CDATA #IMPLIED>
<!ATTLIST BwdJitter Avg CDATA #REQUIRED>

<!ELEMENT FwdLoss EMPTY>
<!ATTLIST FwdLoss Min CDATA #IMPLIED>
<!ATTLIST FwdLoss Max CDATA #IMPLIED>
<!ATTLIST FwdLoss Avg CDATA #REQUIRED>

<!ELEMENT BwdLoss EMPTY>
<!ATTLIST BwdLoss Min CDATA #IMPLIED>
<!ATTLIST BwdLoss Max CDATA #IMPLIED>
<!ATTLIST BwdLoss Avg CDATA #REQUIRED>

<!ELEMENT JLatency EMPTY>
<!ATTLIST JLatency Min CDATA #IMPLIED>
<!ATTLIST JLatency Max CDATA #IMPLIED>
<!ATTLIST JLatency Avg CDATA #REQUIRED>

```

DeviceList.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 1999-2001 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ELEMENT DeviceList (Device*)>
<!ATTLIST DeviceList Type (SAA1|SAA2|ICMP|UDP|DNS|HTTP|VoIP|DHCP|TCP|All) "All">
<!ATTLIST DeviceList View CDATA "All">

<!ELEMENT Device (#PCDATA)>
```

SAADeviceList.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 2000 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ELEMENT SAADeviceList (SAADevice*)>
<!ATTLIST SAADeviceList View CDATA "All">

<!ELEMENT SAADevice (#PCDATA)>
<!ATTLIST SAADevice Types CDATA #REQUIRED>
```

ViewList.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 1999 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->

<!ELEMENT ViewList (View*)>

<!ELEMENT View (#PCDATA)>
<!ATTLIST View Id CDATA #REQUIRED>
```

ThresholdAlert.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 2001 Cisco Systems, Inc.-->
<!-- All Rights Reserved -->
<!-- DTD For the trap response -->

<!ELEMENT ThresholdAlert (SAAEntryNumber?, DevicePair, Threshold, SLCSLA+)>
<!ATTLIST ThresholdAlert SLAType (ICMP|UDP|DNS|HTTP|VOIP|TCP|DHCP) #REQUIRED>
<!ELEMENT SAAEntryNumber (#PCDATA)>
<!ELEMENT DevicePair EMPTY>
<!ATTLIST DevicePair Source CDATA #REQUIRED>
<!-- Target could be either a device hostname or IP -->
<!ATTLIST DevicePair Target CDATA #REQUIRED>
<!-- Threshold is in milliseconds -->
<!ELEMENT Threshold (#PCDATA)>
```

```
<!--ATTLIST Threshold Type (HourlyLatency) "HourlyLatency">
<!--ELEMENT SLCSLA EMPTY>
<!--ATTLIST SLCSLA SlcHandle CDATA #REQUIRED>
<!--ATTLIST SLCSLA SlcName CDATA #REQUIRED>
<!--ATTLIST SLCSLA SlaHandle CDATA #REQUIRED>
<!--ATTLIST SLASLA SlaName CDATA #REQUIRED>
```

Appendix B SLM Error Code Format

Error codes are returned in an XML buffer that conforms to the SlamError DTD. This result is used to flag errors in carrying out the requested operation. The format of the XML buffer is as follows:

```
<SlamError Code="mnemonic_error_code" Sev="severity_level">
  <Descr "...textual description of the error..." /Descr>
  <Cause "... Optional Help text to further describe the problem..." /Cause>
  <Action "... Optional description of steps the user should take to fix the
  problem..." /Action>
</SlamError>
```

Mnemonic Error Code

The error code has the following format:

Facility – Sub Facility – Error

For example, for an ANI error, we might have SLAM-ANI-PERSISTENT_OBJ_ERROR (note the use of underscore and dashes).

The Facility is always "SLAM". The Sub-Facility indicates which component of SLAM issued the error. The Sub-Facilities are listed below:

Sub-Facility Code	Description
ANI	The Asynchronous Network Interface framework
ADMIN	The administration framework.
REPORT	The reporting framework.

Severity Levels

The severity code indicates the seriousness of the condition under which the message was issued.

Severity Code	Description
INFO	Informational message about an action you should take outside the system or about the effects of the command you have issued.
USAGE	Message about an inappropriate use of the system. For example, you issued an incomplete or unknown command.
WARN	Warning message.
SEV	Severe error.
CAT	Catastrophic error.

SLM Error Catalog

SLM error codes and descriptions are listed in SLM Error Catalog files. These XML files (slam_<sub-facility>_error.xml) are stored under the directory:

<CSCOpX>/lib/classpath/com/cisco/nm/slam/error/xml, where <CSCOpX> is the directory in which the SLM product was installed.

Appendix C SLM Troubleshooting Guide

1. The whole Cisco2000 Server is not responding. What should I do?

- **SOLARIS**
 - Telnet to the machine
 - Become root.
 - Stop daemon manager. `"/etc/init.d/dmgtld stop"`
 - Kill any remaining Essentials processes. Do `"ps -ef | grep CSCO"` and stop the listed processes manually with the "kill" command.
 - Remove or clean up the `/var/adm/CSCOpX/log/daemons.log` file
 - Restart daemon manager: `"/etc/init.d/dmgtld start"`
 - Do `"tail -f /var/adm/CSCOpX/log/daemons.log"` to view the log messages.
 - Wait until the **ani** log messages are displayed before doing any web connections
- **Windows NT**
 - Log in as administrator.
 - Stop the daemon manager service by going to the *Control Panel->Services* or typing : `"net stop CRMdmgtld"` at the NT command line prompt.
 - Clean up the logs file in the "`<CSCOpX>/logs`", where "`<CSCOpX>`" is the directory in which the product was installed.
 - Restart the daemon manager service by going to the *Control Panel->Services* or typing `"net start CRMdmgtld"` at the NT command line prompt.
 - If you have **mks** installed, type `"tail <CSCOpX>/logs/SlamServer.log"`, where "`<CSCOpX>`" is the directory in which the product was installed. Wait until the **ani** log messages are displayed before doing any web connections.

2. The SLM server is not responding. What should I do?

- **SOLARIS**
 - Telnet to the machine
 - Become root.
 - Stop the SLM server by typing `"/opt/CSCOpX/bin/pdterm SlamServer"` (SlamServer is case sensitive)
 - Start the SLM server by typing `"/opt/CSCOpX/bin/pdexec SlamServer"`
 - Wait a few minutes for the processes to start up (5 minutes)
- **Windows NT**
 - Log in as administrator
 - Open up a dos command prompt
 - Stop the SLM server. Type: `"pdterm SlamServer"` (SlamServer is case sensitive)

- Start the SLM server. Type: *"pdexec SlamServer"*
- Wait a few minutes for the processes to start (5 minutes)

3. My Java Server Pages (JSP) is not responding. What should I do?

- **SOLARIS**
 - Telnet to the machine.
 - Become root.
 - Stop JRunProxyServer by typing *"/opt/CSCOpX/bin/pdterm JrunProxyServer"* (JRunProxyServer is case sensitive)
 - Start JRunProxyServer by typing *"/opt/CSCOpX/bin/pdexec JRunProxyServer"*
 - Wait a few minutes for the processes to start.
- **Windows NT**
 - Log in as administrator
 - Open up a dos command prompt
 - Stop JrunProxyServer by typing: *"pdterm JrunProxyServer"*
 - Start JrunProxyServer by typing: *"pdexec JrunProxyServer"*
 - Wait a few minutes for the processes to start.

Appendix D Base64 Encoding Example

The following is a public domain code example for generating Base 64 encoded text. This example could be used to generate encoded passwords for use with the SLM Authentication system.

```
// Base64Encoder.java
// $Id: Base64Encoder.java,v 1.3 1997/01/06 12:09:35 abaird Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
/*
                                Copyright NOTICE
                                NOTICE

COPYRIGHT 1995 BY: MASSACHUSETTS INSTITUTE OF TECHNOLOGY (MIT), INRIA

This W3C software is being provided by the copyright holders under the
following license. By obtaining, using and/or copying this software, you
agree that you have read, understood, and will comply with the following
terms and conditions:

Permission to use, copy, modify, and distribute this software and its
documentation for any purpose and without fee or royalty is hereby granted,
provided that the full text of this NOTICE appears on ALL copies of the
software and documentation or portions thereof, including modifications,
that you make.

THIS SOFTWARE IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO
REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT
NOT LIMITATION, COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES OF
MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE
SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS,
COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. COPYRIGHT HOLDERS WILL BEAR NO
LIABILITY FOR ANY USE OF THIS SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising
or publicity pertaining to the software without specific, written prior
permission. Title to copyright in this software and any associated
documentation will at all times remain with copyright holders.
*/

package com.cisco.nm.cmf.security;

import java.io.* ;

/**
 * BASE64 encoder implementation.
 * This object takes as parameter an input stream and an output stream. It
 * encodes the input stream, using the BASE64 encoding rules, as defined
 * in <a href="http://ds.internic.net/rfc/rfc1521.txt">MIME specification</a>
 * and emit the resulting data to the output stream.
 * @see w3c.tools.codec.Base64Decoder
 */
```

```

public class Base64Encoder {
private static final int BUFFER_SIZE = 1024 ;
private static byte encoding[] =
{
'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',          // 0-7
'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',          // 8-15
'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',          // 16-23
'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',          // 24-31
'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',          // 32-39
'o', 'p', 'q', 'r', 's', 't', 'u', 'v',          // 40-47
'w', 'x', 'y', 'z', '0', '1', '2', '3',          // 48-55
'4', '5', '6', '7', '8', '9', '+', '/',          // 56-63
'='                                              // 64
};

InputStream in = null ;
OutputStream out = null ;
boolean      stringp = false ;

private final int get1(byte buf[], int off) {
return (buf[off] & 0xfc) >> 2 ;
}

private final int get2(byte buf[], int off) {
return ((buf[off]&0x3) << 4) | ((buf[off+1]&0xf0) >>> 4) ;
}

private final int get3(byte buf[], int off) {
return ((buf[off+1] & 0x0f) << 2) | ((buf[off+2] & 0xc0) >>> 6) ;
}

private static final int get4(byte buf[], int off) {
return buf[off+2] & 0x3f ;
}

/**
 * Process the data: encode the input stream to the output stream.
 * This method runs through the input stream, encoding it to the output
 * stream.
 * @exception IOException If we weren't able to access the input stream or
 *       the output stream.
 */

public void process ()
throws IOException
{
byte buffer[] = new byte[BUFFER_SIZE] ;
int  got      = -1 ;
int  off      = 0 ;
int  count    = 0 ;
while ((got = in.read(buffer, off, BUFFER_SIZE-off)) > 0) {
if ( got >= 3 ) {
got += off;

```

```

off = 0;
while (off + 3 <= got) {
int c1 = get1(buffer,off) ;
int c2 = get2(buffer,off) ;
int c3 = get3(buffer,off) ;
int c4 = get4(buffer,off) ;
switch (count) {
case 73:
out.write(encoding[c1]);
out.write(encoding[c2]);
out.write(encoding[c3]);
out.write ('\n') ;
out.write(encoding[c4]) ;
count = 1 ;
break ;
case 74:
out.write(encoding[c1]);
out.write(encoding[c2]);
out.write ('\n') ;
out.write(encoding[c3]);
out.write(encoding[c4]) ;
count = 2 ;
break ;
case 75:
out.write(encoding[c1]);
out.write ('\n') ;
out.write(encoding[c2]);
out.write(encoding[c3]);
out.write(encoding[c4]) ;
count = 3 ;
break ;
case 76:
out.write('\n') ;
out.write(encoding[c1]);
out.write(encoding[c2]);
out.write(encoding[c3]);
out.write(encoding[c4]) ;
count = 4 ;
break ;
default:
out.write(encoding[c1]);
out.write(encoding[c2]);
out.write(encoding[c3]);
out.write(encoding[c4]) ;
count += 4 ;
break ;
}
off += 3 ;
}
// Copy remaining bytes to beginning of buffer:
for ( int i = 0 ; i < 3 ;i++)
buffer[i] = (i < got-off) ? buffer[off+i] : ((byte) 0) ;
off = got-off ;

```

```

    } else {
    // Total read amount is less than 3 bytes:
    off += got;
    }
}

// Manage the last bytes, from 0 to off:
switch (off) {
case 1:
out.write(encoding[get1(buffer, 0)]) ;
out.write(encoding[get2(buffer, 0)]) ;
out.write('=') ;
out.write('=') ;
break ;
case 2:
out.write(encoding[get1(buffer, 0)]);
out.write(encoding[get2(buffer, 0)]);
out.write(encoding[get3(buffer, 0)]);
out.write('=');
}
return ;
}

/**
 * Encode the content of this encoder, as a string.
 * This methods encode the String content, that was provided at creation
 * time, following the BASE64 rules, as specified in the rfc1521.
 * @return A String, reprenting the encoded content of the input String.
 */

public String processString () {
if ( ! stringp )
throw new RuntimeException (this.getClass().getName()
+ "[processString]"
+ "invalid call (not a String)");
try {
process() ;
} catch (IOException e) {
}
return ((ByteArrayOutputStream) out).toString() ;
}

/**
 * Create a new Base64 encoder, to encode the given string.
 * @param input The String to be encoded.
 */

public Base64Encoder (String input) {
byte bytes[] = new byte[input.length()] ;
input.getBytes (0, bytes.length, bytes, 0) ;
this.stringp = true ;
this.in      = new ByteArrayInputStream(bytes) ;
this.out     = new ByteArrayOutputStream () ;
}

```

```

/**
 * Create a new Base64 encoder, encoding input to output.
 * @param in The input stream to be encoded.
 * @param out The output stream, to write encoded data to.
 */

public Base64Encoder (InputStream in, OutputStream out) {
    this.in      = in ;
    this.out     = out ;
    this.stringp = false ;
}

/**
 * Testing the encoder.
 * Run with one argument, prints the encoded version of it.
 */

public static void main (String args[]) {
    if ( args.length != 1 ) {
        System.out.println ("Base64Encoder <string>" ) ;
        System.exit (0) ;
    }
    Base64Encoder b = new Base64Encoder (args[0]) ;
    System.out.println ("["+b.processString()+"]" ) ;
    // joe:ej -> am9lOmVvag==
    // 12345678:87654321 -> MTIzNDU2Nzg6ODc2NTQzMjE=
}
}

```