



Cisco TelePresence Supervisor MSE 8050 Remote Management API Reference Guide

D14665.04

December 2010

Contents

Introduction	3
HTTP and HTTPS	3
Consider API overhead when writing applications.....	3
Protocol overview	4
Authentication	4
Message flow	4
Unicode support	6
HTTP headers.....	6
XML header.....	6
Common message elements	6
Authentication	6
Enumerate functions	7
system.xml	7
API reference	9
auditlog.delete	10
auditlog.query.....	10
chassis.alarms.clear.....	10
chassis.alarms.query	10
chassis.blades.query.....	11
chassis.fantrays.query	12
chassis.powershell.query.....	13
chassis.supplyvoltage.query	14
device.health.query	15
device.network.query	15
device.query	17
device.restartlog.query.....	17
Fault codes	18
HTTP keep-alives	20
References	21

Introduction

This reference guide contains the specification of the Cisco TelePresence Supervisor MSE 8050 Remote Management API, by which it is possible to control the Cisco TelePresence Supervisor MSE 8050 blade.

This is accomplished via messages sent using the XML-RPC protocol. XML-RPC is a simple protocol for remote procedure calling using HTTP as the transport and XML as the encoding. It is designed to be as simple as possible, whilst allowing complex data structures to be transmitted, processed and returned. XML-RPC has no platform or software dependence and was chosen over SOAP because of its simplicity.

The interface is stateless. Currently, there is no mechanism for the Cisco TelePresence Supervisor MSE 8050 (Supervisor) to call back the controlling application and therefore the controlling application must poll the Supervisor for status, as required. A future enhancement *may* provide a mechanism for signaling device status changes to the controlling application.

In this implementation of XML-RPC all parameters and return values are part of a <struct> and are explicitly named. For example, the device.query call returns the current time value as a structure member named 'currentTime' rather than as a single value of type <dateTime.iso8601>.

Note: Unless otherwise stated, assume string length is a maximum of 32 characters.

For further details of XML-RPC refer to the [specification](#) [1].

HTTP and HTTPS

The Supervisor expects to receive HTTP communication over TCP/IP connections to port 80. The HTTP messages should be "POST"s to the URL "/RPC2".

HTTPS (a secure, encrypted version of HTTP) is supported on all Supervisors from software version 1.4 and later.

By default, HTTPS is provided on TCP port 443, although Supervisors can be configured to receive HTTP and HTTPS connections on non-standard TCP port numbers if required.

The Supervisor implements HTTP/1.1 as defined by RFC 2616 [2].

Consider API overhead when writing applications

Every API command that your application sends incurs a processing overhead within the Cisco device's own application. The exact amount of overhead varies widely with the command type and the parameters sent. It is important to bear this in mind when designing your application's architecture and software. If the device receives a high number of API commands every second, its overall performance could be seriously impaired – in the same way that it would if several users accessed it from the web interface simultaneously.

For this reason, the best architecture is a single server running the API application and sending commands to the Cisco device. If multiple users need to use the application simultaneously, provide a web interface on that server or write a client that communicates with the server. The server would then manage the clients' requests and send API commands directly to the Cisco device. Implement some form of control in the API application on your server to prevent the Cisco device being overloaded with API commands. This provides much more control than having the clients send API commands directly and will prevent the Cisco device's performance being impaired by unmanageable numbers of API requests.

Protocol overview

Authentication

In order to manage the Supervisor, the controlling application must authenticate itself as a user with relevant privileges. Accordingly, each message contains a user name and password; see the section *Common message elements* below for details of the format. Note that authentication information is sent using plain text and should only be sent over a trusted network.

Note: All calls require administrator privileges.

Message flow

An application can send command messages to the Supervisor. For each command sent (provided the message is correctly formatted according to the [XML-RPC spec](#)), the Supervisor responds with a message indicating success or failure. The response message may also contain any data that was requested.

Command messages are sent in XML format. For example, the following message queries the type of blades in a chassis:

```
POST /RPC2 HTTP/1.1
User-Agent: Frontier/5.1.2 (WinNT)
Host: 10.2.1.100
Content-Type: text/xml
Content-length: 402

<?xml version="1.0"?>
<methodCall>
<methodName>chassis.blades.query</methodName>
<params>
<param>
<value><struct>
<member>
<name>authenticationPassword</name>
<value><string></string></value>
</member>
<member>
<name>authenticationUser</name>
<value><string>admin</string></value>
</member>
</struct></value>
</param>
</params>
</methodCall>
```

A typical response is as follows, some lines have been omitted at the ellipsis (...):

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Content-Length: 240

<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value>
<struct>
<member>
<name>blades</name>
<value>
<array>
<data>
<value>
<struct>
<member>
<name>slot</name>
<value><int>1</int></value>
</member>
<member>
<name>type</name>
<value><string>Cisco TelePresence MSE Supervisor 8050</string></value>
</member>
<member>
<name>softwareVersion</name>
<value><string>2.2(0.7)</string></value>
</member>
<member>
<name>status</name>
<value><string>Blade OK</string></value>
</member>
<member>
<name>portA</name>
<value><string>10.2.132.151</string></value>
</member>
</struct>
</value>
<value>
<struct>
<member>
<name>slot</name>
<value><int>2</int></value>
</member>
<member>
<name>type</name>
<value><string>Cisco Telepresence Server 8710</string></value>
</member>
<member>
<name>softwareVersion</name>
<value><string>2.2(0.26)</string></value>
</member>
</member>
```

```
<name>status</name>
<value><string>Blade OK</string></value>
</member>
<member>
<name>portA</name>
<value><string>10.2.132.152</string></value>
</member>
</struct>
</value>
.....
</member>
</struct>
</value>
</param>
</params>
</methodResponse>
```

The complete list of command messages, their required and optional parameters, and the expected responses are detailed in the sections below. The possible [fault codes](#) are listed later in this document.

Unicode support

Parameters in this version of the API can be in ASCII text or unicode (UTF-8). In order to distinguish between these encodings, any of several methods can be used. If no method is present, ASCII is assumed.

HTTP headers

There are two different ways of specifying unicode in the HTTP headers; either using "Accept-Encoding: utf-8", or modifying the Content-Type header to read "Content-Type: text/xml; charset=utf-8".

XML header

The `<?xml>` tag is required at the top of each XML file. This API will accept an additional encoding parameter with value UTF-8 for this tag, i.e. `<?xml version="1.0" encoding="UTF-8"?>`.

Common message elements

Authentication

All messages must contain a user name and password as follows:

Parameter	Type	Comments
authenticationUser	String	Name of a user with sufficient privilege for the operation being performed. The name is case sensitive.
authenticationPassword	String	The corresponding user's password. This parameter is ignored if the user has no password set. Note that this differs from the web interface where a blank password must be blank.

Note: All calls require administrator privileges.

Enumerate functions

Due to the potential for a very large number of responses, all enumerate functions return an enumerateID response. This contains a value which should be passed to subsequent calls of the same enumerate function in order to retrieve the remainder of the values.

The use of this parameter is as follows:

1. The client computer sends an enumerate call with any necessary parameters (e.g. operationScope) and no enumerateID parameter.
2. The Supervisor returns with an array containing the requested data, and possibly a new enumerateID.
3. If there is an enumerateID, the client will call the enumerate method again, with any parameters that are required or desired, and an enumerateID parameter containing the ID returned by the Supervisor from the previous call. This should be repeated while the Supervisor continues to provide new enumerateID values in responses.
4. After all data is returned, the Supervisor will reply with all remaining results, but no enumerateID.

This method should only be called using enumerateID values as provided by the Supervisor.

system.xml

While not strictly part of the XML-RPC API, some information can be retrieved from the system.xml file. This can be downloaded via HTTP as the file system.xml in the root of the unit, for example <http://TestMCU/system.xml>.

An example is:

```
<system>
  <manufacturer>Cisco</manufacturer>
  <model>Supervisor MSE 8050</model>
  <serial>SM0108E2</serial>
  <chassisSerial>SM010A93</chassisSerial>
  <softwareVersion>2.1(1.18)</softwareVersion>
  <buildVersion>8.5(1.18)</buildVersion>
  <hostName>TestSupervisor</hostName>
  <uptimeSeconds>2345</uptimeSeconds>
</system>
```

The meaning of the fields is described in the table below.

Field	Comments
manufacturer	The manufacturer of this Supervisor, e.g. Cisco
model	The model of this particular Supervisor, e.g. MSE 8050.
serial	The serial number of this Supervisor.

chassisSerial	The serial number of the MSE chassis.
softwareVersion	The software version currently running.
buildVersion	The build version of the software currently running.
hostName	The host name of the system.
uptimeSeconds	The number of seconds since the Supervisor was booted.

API reference

This section is a reference to each of the Supervisor API calls. For each API call, the following information is provided where applicable:

- ▶ Description of the call's function
- ▶ Parameters
- ▶ List of responses
- ▶ Response data types
- ▶ Structure formats
- ▶ Additional information that will assist in using the API call
- ▶ Parameters deprecated from previous versions

The table below lists all currently supported Supervisor API calls. Click on the call name to go to the page containing a complete description of the call.

- ▶ [auditlog.delete](#)
- ▶ [auditlog.query](#)
- ▶ [chassis.alarms.clear](#)
- ▶ [chassis.alarms.query](#)
- ▶ [chassis.blades.query](#)
- ▶ [chassis.fantrays.query](#)
- ▶ [chassis.powershell.query](#)
- ▶ [chassis.supplyvoltage.query](#)
- ▶ [device.health.query](#)
- ▶ [device.network.query](#)
- ▶ [device.query](#)
- ▶ [device.restartlog.query](#)

auditlog.delete

This call is used to delete stored events from the Event log.

Response	Type	Comments
deleteIndex	Integer	You can delete logs in segments of 400 entries. To delete logs, enter the value returned by the deleteableIndex response (see auditlog.query below). This will delete all complete 400 log segments below this amount, leaving the residuals. Alternatively, you can delete less than this amount by picking a number below the value of deleteableIndex. This will delete all complete 400 logs segments below that number, leaving any residuals. Stored audit events up to and including the indicated deleteIndex will be permanently deleted.

auditlog.query

This call is used to request data about the Event log. The call takes no parameters. The response returns the following.

Response	Type	Comments
firstIndex	Integer	The index of the oldest stored audit event.
deletableIndex	Integer	The index of the most recent deletable audit event.
numEvents	Integer	The total number of events stored.
percentageCapacity	Integer	The percentage of the total available capacity being used by the audit log.

chassis.alarms.clear

This call is used to clear historic alarms. It takes no parameters and has no response fields.

chassis.alarms.query

This call is used to request data about the chassis alarms. The call takes no parameters. The response returns the following.

Response	Type	Comments
alarms	Array	Contains alarm information in the structure described below.

The alarms array consists of structures which contain the following fields.

Field	Type	Comments
alarmName	String	Identifies the alarm.
alarmLevel	String	The relative importance of the alarm. One of: None Minor Major Critical
alarmState	String	The current alarm state. One of: OK Alarm Muted HistoricAlarm
alarmTitle	String	A verbose string that identifies alarms that have titles.
alarmDescription	String	A verbose description of the alarm.

chassis.blades.query

This call is used to request data about the blades installed in the chassis. The call takes no parameters. The response returns the following.

Response	Type	Comments
blades	Array	Contains the blade information in structures as described below.

The blades array consists of structures which contain the following fields.

Field	Type	Comments
slot	Integer	Number of the chassis slot that the blade is installed in.
type	String	The type of blade.
softwareVersion	String	Software version of the blade.

Field	Type	Comments
status	String	<p>The status of the blade. One of:</p> <ul style="list-style-type: none"> Blade absent Blade OK Waiting for communications Lost communication Blade software too old Please upgrade Blade shut down Blade shutting down Blade attempting restart Blade restarting <p>Concatenated with any of the following and separated by semicolons:</p> <ul style="list-style-type: none"> Temperature critical Voltages critical Requires restart Blade badly inserted Invalid blade ID RTC Battery critical <p>For example:</p> <ul style="list-style-type: none"> Blade shut down Blade shut down; Requires restart Blade shut down; Requires restart; Temperature critical <p>Note: The status Blade OK is only shown if none of the other statuses apply.</p>
portA	String	The IP address of the port. In the format a.b.c.d.
portB	String	
portC	String	
portD	String	

chassis.fantrays.query

This call is used to request the current fan tray status on the chassis. It takes no parameters and has the following response fields.

Response	Type	Comments
fanTrays	Array	Contains fan tray information in structures as described below.

The fanTrays array consists of structures which contain the following fields.

Field	Type	Comments
tray	String	Identifies the fan tray either Upper or Lower.
status	String	The status of the fan tray. One of: OK Problem Initializing Fan tray absent
type	String	Model name of the fan tray. Only appear if the status is OK or Problem.
serial	String	Serial number of the fan tray. Only appear if the status is OK or Problem.
averageFanSpeed	Integer	Average speed of the fan in RPM. Only appear if the status is OK or Problem.

chassis.powershell.query

This call is used to query the current supply voltage status on the chassis. It takes no parameters and has the following response fields.

Response	Type	Comments
powerShelves	Array	Contains power shelf information in a structure as described below.

The powerShelves array consists of structures which contain the following fields.

Field	Type	Comments
shelf	String	Identifies the power shelf, either A or B.
status	String	The status of the power shelf. One of: OK Power shelf reporting fault Power shelf not monitored Lost contact with power shelf Insufficient current capacity Authentication failed
type	String	The model of the power shelf. This field only appears when the status is other than Authentication failed and Valere monitoring is enabled on the web interface.
reportedInfo	Array	Contains information for each rectifier in a structure as described below.

The reportedInfo array consists of structures which contain the following fields. This structure doesn't appear if power shelf monitoring is disabled on the web interface or the authentication has failed.

Field	Type	Comments
rectifier	Integer	Identifies which Valere rectifier this information is for. 1 to 4.
status	String	The status of the Valere rectifier. This is reported directly by the rectifier.
voltage	Integer	Voltage level in mV.
current	Integer	Current level in mA.
capacity	Integer	Current capacity in mA.

chassis.supplyvoltage.query

This call is used to query the current supply voltage status on the chassis. It takes no parameters and has the following response fields.

Response	Type	Comments
powerSupplies	Array	Contains power supply information in structure described below.

The powerSupplies array consists of structures which contain the following fields.

Field	Type	Comments
supply	String	Identifies which power supply that was queried. Either A or B.
status	String	The status of the power supply voltage. One of: OK Out of range high Out of range low Too high Too low Supply not monitored
voltage	Integer	Voltage level in mV. This field is absent if the status is Out of range low or Supply not monitored.

device.health.query

Returns the current status of the Supervisor, such as health monitors and CPU load.

Response	Type	Comments
cpuLoad	Integer	The CPU load, as a percentage.
temperatureStatus	String	One of: ok outOfSpec critical.
temperatureStatusWorst	String	
rtcBatteryStatus	String	
rtcBatteryStatusWorst	String	
voltagesStatus	String	
voltagesStatusWorst	String	
operationalStatus	String	One of: active shuttingDown shutDown.

device.network.query

This call takes no parameters. The response returns the following:

Parameter	Type	Comments
portA	Array (see below)	Contains the configuration and status for port A.
portB	Array (see below)	Contains the configuration and status for port B.

The format for the two structures above is:

Field	Type	Comments
enabled	Boolean	True if the port is enabled, otherwise false.
linkStatus	Boolean	True if the link is up, false if the link is down.
speed	Integer	Speed of the link from the port in Mbps. One of: 10 100 1000
fullDuplex	Boolean	True if full duplex enabled, false if half duplex enabled.

Field	Type	Comments
macAddress	String	A 12 character string with no separators.
packetsSent	Integer	Statistics from the web interface. It is worth noting that all these values are 32 bit signed integers, and thus may wrap.
packetsReceived	Integer	
multicastPacketsSent	Integer	
multicastPacketsReceived	Integer	
bytesSent	Integer	
bytesReceived	Integer	
queueDrops	Integer	
collisions	Integer	
transmitErrors	Integer	
receiveErrors	Integer	
bytesSent64	String	64 bit versions of the above statistics, using a string rather than an integer.
bytesReceived64	String	
Optional parameters		
hostName	String	The host name of this port.
dhcp	Boolean	True if configured by DHCP, otherwise false.
ipAddress	String	The IP address of the Supervisor in a.b.c.d format.
subnetMask	String	The subnet mask address of the Supervisor in a.b.c.d format.
defaultGateway	String	The default gateway address of the Supervisor in a.b.c.d format.
domainName	String	The domain name of this port.
nameServer	String	The address of the name server for this Supervisor in a.b.c.d format.
nameServerSecondary	String	The address of the secondary name server for this Supervisor in a.b.c.d format.

The optional fields above will be returned only if the interface has been enabled and has been configured.

device.query

There are no parameters passed with this method call. The method response returns the following:

Parameter	Type	Comments
currentTime	dateTime.iso8601	The system's current time (UTC).
restartTime	dateTime.iso8601	The date and time at which the system was last restarted.
serial	String	The serial number of the device.
chassisSerial	String	The serial number of the chassis.
softwareVersion	String	The software version of the running software.
buildVersion	String	The build version of the running software.
model	String	The model of this device, e.g. "Cisco TelePresence MSE 8050".
apiVersion	String	The version number of the API implemented by this device.
activatedFeatures	Array	Currently, only contains a string "feature" with a short description of the feature.

device.restartlog.query

This call is used to return the restart log (also known as the system log on the web interface).

Response	Type	Comments
log	Array	Contains the restart log in structures as described below.

The log array consists of structures which contain the following fields.

Field	Type	Comments
time	dateTime.iso8601	The time of the last reboot.
reason	String	The reason for the reboot. One of: unknown User requested shutdown User requested upgrade

Fault codes

The Supervisor shares a series of fault codes with Cisco TelePresence MCUs, gateways and VCRs, the fault codes are returned when a fault occurs during the processing of an XML-RPC request. While individual call descriptions above give some indication of which faults may occur, below is a description of all possible fault codes used within this specification and the most common interpretation. Note that not all codes are used by the Supervisor.

Fault Code	Description
1	Method not supported. This method is not supported on this device.
2	Duplicate conference name. A conference name was specified, but is already in use.
3	Duplicate participant name. A participant name was specified, but is already in use.
4	No such conference or auto attendant. The conference or auto attendant identification given does not match any conference or auto attendant.
5	No such participant. The participant identification given does not match any participants.
6	Too many conferences. The device has reached the limit of the number of conferences that can be configured.
7	Too many participants. There are already too many participants configured and no more can be created.
8	No conference name or auto attendant id supplied. A conference name or auto attendant identifier was required, but was not present.
9	No participant name supplied. A participant name is required but was not present.
10	No participant address supplied. A participant address is required but was not present.
11	Invalid start time specified. A conference start time is not valid.
12	Invalid end time specified. A conference end time is not valid.
13	Invalid PIN specified. A PIN specified is not a valid series of digits.
14	Unauthorised. The requested operation is not permitted on this device.
15	Insufficient privileges. The specified user id and password combination is not valid for the attempted operation.
16	Invalid enumerateID value. An enumerate ID passed to an enumerate method invocation was invalid. Only values returned by the device should be used in enumerate methods.
17	Port reservation failure. This is in the case that reservedAudioPorts or reservedVideoPorts value is set too high, and the device cannot support this.

Fault Code	Description
18	Duplicate numeric ID. A numeric ID was given, but this ID is already in use.
19	Unsupported protocol. A protocol was used which does not correspond to any valid protocol for this method. In particular, this is used for participant identification where an invalid protocol is specified.
20	Unsupported participant type. A participant type was used which does not correspond to any participant type known to the device.
21	No such folder. A folder identifier was present, but does not refer to a valid folder.
22	No such recording. A recording identifier was present, but does not refer to a valid recording.
23	No changes requested. This is given when a method for changing something correctly identifies an object, but no changes to that object are specified.
24	No such port. This is returned when an ISDN port is given as a parameter which does not exist on an ISDN gateway.
101	Missing parameter. This is given when a required parameter is absent. The parameter in question is given in the fault string in the format "missing parameter - <i>parameter_name</i> ".
102	Invalid parameter. This is given when a parameter was successfully parsed, is of the correct type, but falls outside the valid values; for example an integer is too high or a string value for a protocol contains an invalid protocol. The parameter in question is given in the fault string in the format "invalid parameter - <i>parameter_name</i> ".
103	Malformed parameter. This is given when a parameter of the correct name is present, but cannot be read for some reason; for example the parameter is supposed to be an integer, but is given as a string. The parameter in question is given in the fault string in the format "malformed parameter - <i>parameter_name</i> ".
201	Operation failed. This is a generic fault for when an operation does not succeed as required.

HTTP keep-alives

A method of reducing the amount of TCP traffic when polling the Supervisor via the API is to use HTTP keep-alives).

Any client which supports HTTP keep-alives may include the following line in the HTTP header of an API request:

```
Connection: Keep-Alive
```

This indicates to the Cisco TelePresence product that the client supports HTTP keep-alives. The Supervisor *may* then choose to not close the TCP connection after returning its response to the request. If the connection will be closed, the Supervisor returns the following line in the HTTP header of its response:

```
Connection: close
```

The absence of this line indicates that the Supervisor will keep the TCP connection open and that the client may use the same connection for a subsequent request.

The Supervisor will not allow a connection to be kept alive if:

- ▶ the current connection has already serviced a set number of requests
- ▶ the current connection has already been open for a certain amount of time
- ▶ there are already more than a certain number of connections in a “kept alive” state

These restrictions are in place to limit the resources associated with kept-alive connections. If a connection is terminated for either of the first two reasons, the client will probably find that the connection is back in a keep-alive state following the next request.

The client should never assume a connection will be kept alive.

Also note that, even after a response not containing the “connection: close” header, the connection will still be closed if no further requests are made within one minute. If requests from the client are likely to be this far apart then there is little to be gained by using HTTP keep-alives.

References

The following table lists documents and web sites referenced in this document. All product documentation can be found on our [web site](#).

Title	Link
XML-RPC	http://www.xmlrpc.com/
Hypertext Transfer Protocol (HTTP/1.1)	http://www.faqs.org/rfcs/rfc2616.html

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 NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

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

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at www.cisco.com/go/trademarks. Third party trademarks mentioned 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. (1005R)

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

© December 2010 Cisco Systems, Inc. All rights reserved.