MOTOROLA LABS

TITLE:

User Guide for the Device Configuration System

SOURCE:

Networks and Infrastructure Research Lab

AUTHORS:

Name	Organization
Kabe VanderBaan	Networks and Infrastructure Research Lab, Illinois
Justin Hoffman	MSU Capstone
Jason Ruthkoski	
Steven Ford	
Jae Eun Shin	

DATE:

December 5, 2005
Version 1.11

Revision History

Version	**Date**	**Change Description**
1.0	10/25/2005	User Guide Doc created
1.1	10/30/2005	Added more content
1.2	11/01/2005	Reworded some content, and added more content where needed, added channel diagram
1.3	11/04/2005	Added more content, overhauled Configurable components to describe all XML variables, replaced System Overview diagram
1.4	11/08/2005	Fixed formatting errors, added new System Overview diagram, a Managed Device section, Connections section, and a new connections diagram
1.5	11/10/2005	Executive Summary changed, added Control loop diagram, edited system overview diagram to reflect control loop, fixed configuration files to reflect the new format
1.6	11/14/2005	Fixed formatting errors, edited and resized diagrams, reduced spacing in XML files.
1.7	11/15/2005	Reworded some content, resized control loop diagram, edited system overview diagram, added examples to many of the configuration components.
1.8	11/17/2005	Edited and resized diagrams.
1.9	11/21/2005	Added Graphical User interface and Execution section.
1.10	11/28/2005	Edited GUI, configuration contents, and the reference.
1.11	12/05/2005	Inserted screen captures for GUI tasks, and organized the references.

1 Executive Summary

This document describes how to configure and use the Device Configuration System (DCS). DCS is designed and developed as a universal configuration server, able to communicate with and configure multiple heterogeneous devices, hiding the configuration complexities away from the user.

DCS creates a control loop that receives state updates of a managed device in the form of events. It then analyzes the current state of the device to determine if the device is in an undesired state, and will submit a solution to the device to transition it back into a desired state.

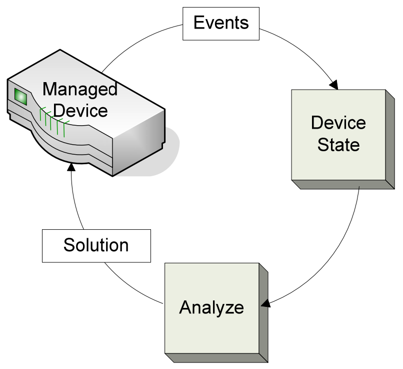


Figure 1: The DCS control loop

In DCS, each device has a proxy that acts as an intermediary to communicate between the managed device and the configuration server. Using the proxy, the configuration server is capable of interacting with any manageable device; however the target devices are for network infrastructure. The server contains a list of commands and command dependencies per device. Events are generated from a device when a change of states occurs. The event is sent to the proxy, which translates and sends the event to the server. The server identifies which device sent the event by a manufacturer/ model number / operating system version number tuple (i.e. a Cisco 2950T running ios 12.1(22)EA4 is represented as Cisco/2950T/IOS-12.1(22)EA4). In response to an event, the configuration server will analyze the current state of the device. If the server determines the device is in an undesired state, the server selects a set of commands (including command dependencies) and submits the commands to the proxy which in turns executes the commands on the device. The commands will transition the device to a desired state.

The major points of this system are as follows:

1) A proxy can communicate with a managed device using Simple Network Management Protocol (SNMP), Secure Shell (SSH), telnet, or serial line.
2) A server maintains and updates the state of a device by capturing alerts/events from the managed device via a proxy.
3) The proxy and the server communicate using Remote Method Invocation (RMI) or Simple Object Access Protocol (SOAP).
4) The server can request additional state information from the device create a solution.
5) The server is able to determine when the device is in an undesired state.
6) The server submits a solution to transition the device into a desired state to the proxy.
7) The proxy executes the solution on the managed device.

2 System Overview

The high-level architecture for DCS consists of one or more managed devices, a managed device proxy and a configuration server. The managed device and the proxy communicate over two protocols, SNMP, and CLI. The proxy and server communicate over three connections, event, command, and control. The system manages the state of the managed device. When the event is received from the managed device, the configuration server analyzes the current state of the device to determine if the device is in an undesired state. If the managed device is in an undesired state, the configuration server will submit a solution to the device to transition the device back into a desired state.

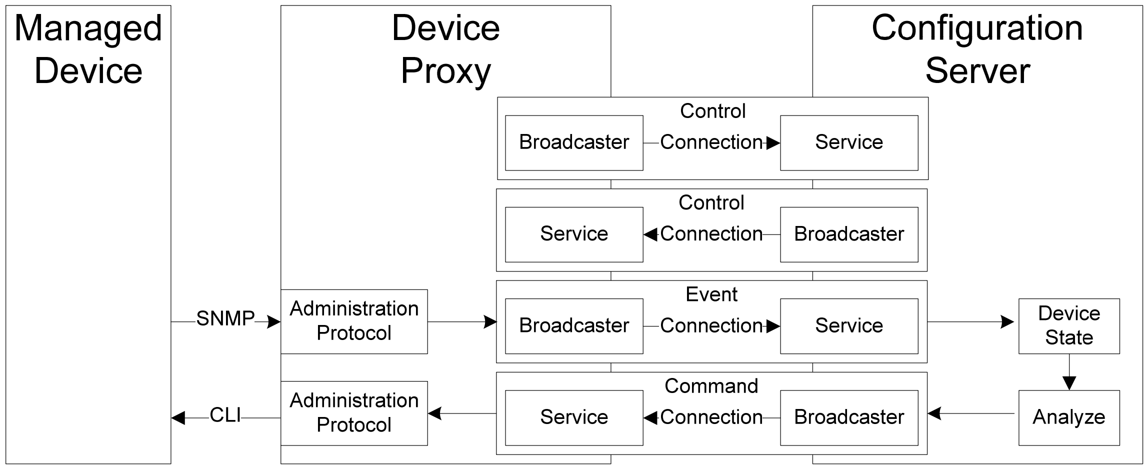


Figure 2: High-level DCS architecture

2.1 Managed Device

A managed device is any device on a network that needs to be managed. This will typically be a network device such as a router or switch.

2.2 Configuration Server

The configuration server is the core of the system. Ultimately, the server maintains and analyzes the state of a managed device. It recognizes that there are problems with the managed device and performs the appropriate actions to remedy the situation. It uses three connections to communicate with a device proxy event, command, and control (see 2.4). These connections ultimately create a control loop in which the event connection is used to inform the server of state changes on the device, and the command connection is used to issue commands back to the device to return the device to a desired state. The control connection maintains the control loop.

2.2.1 Device State

The device state is the current status of the device. There are two states in the DCS and they are undesired state and desired state. The state of the managed device is stored and monitored by the configuration server as a model, and the model values are updated by the alert. Alert is an event from the proxy that contains the changing state information of the device. If the device state changes to an undesired state, the configuration server will submit the solution to the device to put it back to a desired state.

2.2.2 Analyze

In order to submit the solution, the server must analyze the device state. Configurable alerts are setup by system administrators that will indicate to the server that an undesired state has been reached. Remedies can be associated with alerts in the server's rule configuration. When a particular alert is fired, the associated remedy will there be sent to the device.

2.3 Device Proxy

A device proxy allows the server to communicate with the managed device. The proxy receives events and data from the device and translates them for the server to understand. Additionally, the proxy receives commands from the server that it must execute on the device. The proxy talks to the device using the native interfaces and protocols available on the device (see 2.3.1). Currently, SNMP, secure shell (SSH), telnet, and serial line are supported for device to proxy communications.

The device proxy uses three connections to communicate with the configuration server, event, command, and control. These connections ultimately create a control loop in which the event connection is used to inform the server of a problem on a device, and the command connection is used to issue commands back to the device to fix the problem. The control connection is used to maintain the control loop.

2.3.1 Administration Protocols

The device proxy uses administration protocols to communicate with a registered device. Currently there are two supported administration protocols, command line interface (CLI) and Simple Network Management Protocol (SNMP). A CLI is used to send commands from a proxy to a device. Supported CLIs include Secure Shell (SSH), Telnet, and Serial. SNMP is used to receive events from a managed device (see 3.1.2.2).

2.4 Connections

The server and the proxy communicate over connections. The three types of connections are event, command, and control. A connection consists of three parts, a broadcaster (the source), a service (the sink) and a channel. There are 2 supported channels, Java Remote Method Invocation (RMI) and Simple Object Access Protocol (SOAP). The following diagram shows the components that make up the connections (see 3.1.1 for more information).

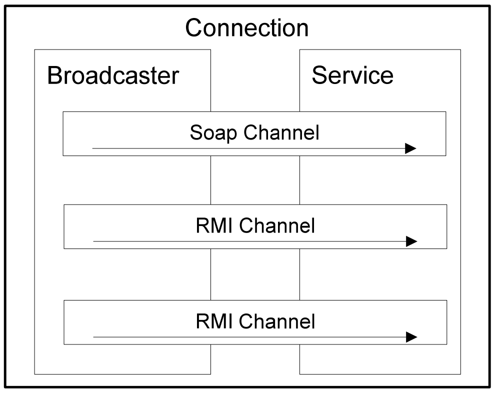


Figure 3: A connection that has one SOAP channel and two RMI channels.

2.4.1 Services

Services are the receiving side of the connection. Services accept messages using any supported channel (i.e., RMI, SOAP).

2.4.2 Broadcasters

Broadcasters are the sending side of the channel. Broadcasters send a message using the first available channel (i.e., RMI, SOAP).

2.4.3 Channels

The actual communication between the configuration server and proxy is done through message channels. There are two different channels available, one using Java Remote Method invocation (RMI) and one using Simple Object Access Protocol (SOAP). Both protocols can be used at the same time.

2.4.4 Event Connection

The event connection allows a proxy to send device state updates to the server in the form of events that were received from the device. Any time the proxy detects a device state change an event message is sent from the proxy to the server.

2.4.5 Command Connection

The command connection allows a server to send commands to the proxy that need to be executed on the managed device. Command messages are sent from a server to a device describing a command that is to be executed.

2.4.6 Control Connection

The control connection allows the proxy and server to send control messages to each other. A control message contains information about the proxy and a managed device so that the server can manage the control loop for the device. Control messages can be used to register and un-register proxies and devices, and to setup connections between the proxy and the server.

3 Configuration

This section describes how to configure the system. This will describe all the configurable components and configuration files. In order to allow the proxy modules to connect to the server, the connection must be configured. To allow the proxy to talk to a device, administration protocols must be configured. To allow for the management of devices from the server, a set of comments, remedies, alerts, and rules must be in the server configuration. There are four kinds of configuration file need to be defined to manage the device properly. Those are the proxy, server, and other configuration files which are the device, and server configuration file.

3.1 Configurable Components

The sub items in this section describe the configurable components of the system. Configurable components include connections, devices, administration protocols, systems, rules to determine the undesirable state, and log4j.

3.1.1 Connections

Connections are the method by which the server and proxy communicate. This section describes how to configure the connections.

3.1.1.1 Channels

This section describes how to configure each Channel that connects services and broadcasters. There are two sides of a channel, the service side and the broadcast side. The service side is the receiving side and the broadcast side is the sending side. Each side of the channel must configure the channel that is being used. There are two types of channels, Remote Method Invocation (RMI) and Simple Object Access Protocol (SOAP).

To configure an RMI Channel:
```
<RMIChannel host="" port="" name="" />
```

Attribute values:

`host:`	The IP address where the RMI registry will reside.
`port:`	The port on which the RMI channel will communicate.
`name:`	The name of the RMI channel.

EXAMPLE:

```
<RMIChannel host="localhost" port="1099" name="RMIProxyControl"/>
```

To configure a SOAP channel:
```
<SOAPChannel host="" port="" name="" />
```

Attribute values:

`host:`	The IP address where the SOAP registry will reside.
`port:`	The port on which the SOAP channel will communicate.
`name:`	The name of the SOAP channel.

EXAMPLE:

```
<SOAPChannel host="localhost" port="1099" name="SOAPProxyControl"  />
```

3.1.1.2 Global Parameters

This section defines parameters that are global to the proxy, or the server. The parameters that are set affect all of the connections. Global Parameters are same value for all connections. Attributes set for a specific connection are specific only to that connection. A parameter list is used to create a collection of parameters which apply to a particular service.

```
<GlobalParameters>
   <ParameterList name="">
      <Param name="" value=""/>
      <Param name="" value=""/>
   </ParameterList>
</GlobalParameters>
```

Attribute values:

`name:`	The name of the parameter list.
`Param name:`	The name of the actual parameter.
`value:`	The value of the parameter.

Currently, global parameters only need to be set when using SOAP channels. Only one sub-system (`SystinetWasp` for web service channels) makes use of `GlobalParameters`. Web Applications and Services Platform (WASP) is a protocol for reliable messaging over SOAP. If there is no `ParameterList` with the name `SystinetWasp`, then web service channels cannot be configured or used.

Systinet Wasp parameters:

`home:`	The home directory of Systinet Wasp.
`port:`	The TCP service port for listening.
`max-threads:`	The maximum number of threads to allow within Systinet Server.
`max-readtime:`	The maximum number of milliseconds that can be spent reading.
`debug-level:`	The level of output to put into the Systinet log files.

Example:

```
<GlobalParameters>
   <ParameterList name="SystinetWasp">
      <Param name="home"         value="wasp"/>
      <Param name="port"         value="8080"/>
      <Param name="max-threads"  value="50"/>
      <Param name="max-readtime" value="10000"/>
      <Param name="debug-level"  value="0"/>
   </ParameterList>
</GlobalParameters>
```

3.1.1.3 Services

Services are the receive side of a connections. There is one service per connection. There are three types of services, event, command, and control.

```
<Service name="">
   [Channel]
   [Channel]
</Service>
```

Attribute values:

`name:`	The name of the channel. Possible values are `control, event, and command`.
`[Channel]:`	The Channel to communicate over (See 3.1.1.1).

3.1.1.4 Broadcasters

Broadcasters are the send side of a channel. There is one broadcaster per connection. There are 3 types of broadcasters, event, command, or control.

```
<Broadcaster name="">
   [Channel]
   [Channel]
</Broadcaster>
```

Attribute values:

`name:`	The name of the channel. Possible values are `control, event, and command`.
`[Channel]:`	The Channel to communicate over (See 3.1.1.1).

3.1.2 Devices

This section describes how to configure devices on the server and the proxy. The server and the proxy use different formats for the device configuration. In the proxy configuration, a device signifies what device the proxy is managing. In the server configuration, it contains a list of the device files corresponding to each type of device that is supported by the system.

For the proxy:

```
<Device manufacturer="" model="" system="">
```

Attribute values:

`manufacturer:`	The manufacturer of the device. For example, Cisco.
`model:`	The device model. For example, 2950T.
`system:`	The name of operating system that the device uses. For example, IOS-12.1(22)EA4.

EXAMPLE:

```
<Device manufacturer="Cisco" model="2950T" system="IOS-12.1(22)EA4">
```

For the Server:

```
<Devices>
 <Device source="" />
</Devices>
```

<u>Attribute Values:</u>

`source:`	The pathname of the device configuration file.

EXAMPLE:

```
<Devices>
      <Device source="conf\devices\cisco.2950t.xml" />
      <Device source="conf\devices\intel.x86.xml" />
</Devices>
```

3.1.2.1 Model Variables

Model Variables are used to initialize the state of a device. This should only be used for state that cannot be accessed over the management interfaces, or to bootstrap information.

```
<ModelVariables>
   <Value key="" value=""/>
   <Value key="" value=""/>
</ModelVariables>
```

<u>Attribute values:</u>

`key:`	The name of the variable.
`value:`	The value associated with that variable.

<u>EXAMPLE:</u>

```
<ModelVariables>
   <Value key="hostname" value="localhost"/>
   <Value key="password" value="*MotoDemo06*"/>
</ModelVariables>
```

3.1.2.2 Administration Protocols

A managed device proxy uses Administration protocols to communicate with a registered device. There are two types of protocols, CLI and SNMP. There are three CLI protocols and one SNMP protocol. The three CLI protocols are Telnet, SSH, and Serial.

```
<AdminProtocols>
   [Protocol]
</AdminProtocols>
```

<u>Attribute values:</u>

`[Protocol]:`	The protocol being used (see 3.1.2.2.1 and 3.1.2.2.2).

3.1.2.2.1 *Simple Network Management Protocol (SNMP)*

A SNMP administration protocol is used by a proxy to receive events from a managed device. There are three versions of SNMP, and each is configurable. Version 1 and 2 are similar and therefore are configured the same, while version 3 is a major overhaul of the protocol adding a number of security features, and therefore more configurable options.

The library used by DCS is SNMP4_13 by Westhawk ①.

To configure SNMP version 1 and 2:

```
<SNMP
   version=""
   host=""
   port=""
   debug-level=""
   socket-type=""
   community=""
/>
```

Attribute values:

`version:`	The SNMP version number. Possible values are 1, 2, and 3.
`host:`	The host name of the machine that we want to receive events from.
`port:`	The UDP service port number which will receive SNMP traps.
`debug-level:`	The level of debugging that will be done. 0 is no debug, 1 is fatal, 2 is critical, and 3 is non-critical.
`socket-type:`	The type of socket that is used. By default the “Standard” socket will be used. The “Netscape” socket provides extra functionality for the Netscape capabilities classes.
`community:`	Name of the SNMP community to receive traps from (this is only valid for v1, v2).

EXAMPLE:

```
<SNMP version="2" host="192.168.2.254" port="162" debug-level="2"
socket-type="Standard" community="public"/>
```

To configure SNMP version 3:

```
<SNMP
   version = ""
   host=""
   port=""
   socket-type=""
   user = ""
   user-privflag = ""
   user-authflag = ""
   auth-proto = ""
   auth-password = ""
   priv-password = ""
   context-name = ""
   context-engineid = ""
/>
```

Attribute values:

`version:`	The SNMP version number. Possible values are 1, 2, and 3.
`host:`	The host name of the machine that we want to receive events from.
`port:`	The UDP service port number which will receive SNMP traps.
`socket-type:`	The type of socket that is used. By default the "Standard" socket will be used. The "Netscape" socket provides extra functionality for the Netscape capabilities classes.
`user:`	The user name of the SNMP user profile. It specifies the user on whose behalf the message is being exchanged. By default, the value is "initial".
`user-authflag:`	Whether the user wants to use authentication or not. If the user wants authentication the value is "True". If not, the value is "False". Default is "False".
`auth-password:`	The user authentication password.
`auth-proto:`	The authentication protocol the user wants to use (as an integer). The available options are 0 and 1. MD5 is 0, and SHA1 is 1. The default is 0.
`user-privflag:`	Whether the user wants to use privacy or not, by default false.
`priv-password:`	The user privacy password.
`context-engineid:`	The context engine id of the host.
`context-name:`	The context name of the host. It could be left empty (i.e. "").

EXAMPLE:

```
<SNMP version="3" host="192.168.2.254" port="162"
                  socket-type="Standard" user="public" />
```

3.1.2.2.2 Command Line Interface (CLI)

CLI protocol is used from proxy to execute commands on the device. The "`cli`" protocols can be Serial, SSH, or Telnet.

3.1.2.2.2.1 Telnet

The Telnet protocol is used for a proxy to connect to the device via Telnet. In order to use the Telnet protocol, host, port, user, and password information must be defined.

To configure Telnet:

```
   <Telnet
      host=""
      port=""
      user=""
      password=""/>
```

Attribute values:

`host:`	The address of the device.
`port:`	The port number for telnet server of the device. The default value is 23.
`user:`	The username to connect to the telnet server of the device. Leave empty ("") if the user name is not required.
`password:`	The password to connect to the telnet server of the device. Leave empty ("") if the password is not required.

EXAMPLE:

```
<Telnet host="192.168.2.254" port="23" user="" password="*MotoDemo05*"/>
```

3.1.2.2.2.2 Secure Shell

A Secure Shell (SSH) administration protocol is used for a proxy to connect to the managed device by SSH. SSH uses encryption. In order to use SSH protocol, host, port, user, and password information have to be defined.

To configure SSH:

```
   <SSH
      host=""
      port=""
      user=""
      password=""/>
```

<u>Attribute values:</u>

`host:`	The address of the device.
`port:`	The port number for SSH server of the device. Default value is 22.
`user:`	The username to connect to the SSH server of the device. Leave empty ("") if the user name is not required.
`password:`	The password to connect to the SSH server of the device. Leave empty ("") if the password is not required.

EXAMPLE:

```
<SSH host="35.9.26.64" port="22"
      user="demouser" password="*MotoDemo05*" />
```

3.1.2.2.2.3 Serial Protocol

The Serial protocol is used when the device is connected to the proxy using a serial line.

```
   <Serial
      portName=""
      timeout=""
      baudrate=""
      dataBits=""
      stopBits=""
      parity="" />
```

<u>Attribute values:</u>

`portName:`	The port name of a serial line.
`timeout:`	The value of timeout. Default is 2000.
`baudrate:`	The value of the `baudrate` for serial line. Default is 9600.
`databits:`	The value of the `dataBits` for serial line. Default is 8.
`stopBits:`	The value of the `stopBits` for serial line. Default is 1.
`parity:`	The value of `parity` for the serial line. Default is 0.

EXAMPLE FOR WINDOWS:

```
<Serial portName="COM1" timeout="2000" baudrate="9600" dataBits="8"
          stopBits="1" parity="0"/>
```

EXAMPLE FOR UNIX:

```
<Serial portName="/dev/term/a" timeout="2000" baudrate="9600"
          dataBits="8" stopBits="1" parity="0"/>
```

3.1.3 Systems

The systems section is used in the server configuration to have commands for a particular operating system. It lists out the file names for each operating system that is available to manage (see 3.3).

```
<Systems>
  <System source= "" />
</Systems>
```

Attribute values

`source:`	The pathname of the system configuration file.

EXAMPLE:

```
<Systems>
  <System source="conf\systems\ios-12.1.22.ea4.xml" />
  <System source="conf\systems\windows-xp-pro-sp2.xml" />
</Systems>
```

3.1.4 Log4j

The log4j is used to log debug and error messages for DCS and any proxy modules. Since Log4j configuration modules, the use of xml name spaces the `xmlns:log4j="://jakarta.apache.org/log4j/"` attribute is required on the root element of the server and proxy configuration files. The documentation on log4j configuration is available here ②.

The log4j configuration has following format:

```
<log4j:configuration>
   <appender name="" class="">
     <param name="" value=""/>
     <layout class="">
        <param name="" value="" />
     </layout>
   </appender>
   <category name="">
     <priority value="" />
     <appender-ref ref="" />
   </category>
   <root>
      <priority value="" />
      <appender-ref ref="" />
   </root>
```

```
</log4j:configuration>
```

<u>Attribute values</u>

`appender name:`	The name of the appender.
`class:`	The name of the class that this appender uses.
`name:`	The name of the parameter. “File” and “Append” param name needs to be defined for the appender.
`value:`	The actual value to the name of the parameter. The path of the file for name “File” and the boolean value for the name “Append” param need to be defined.
`layout class:`	The name of the class that this layout uses.
`category name:`	The category name this log4j uses.
`priority value:`	The value is “debug”.
`appender-ref ref:`	The name of the appender for this log4j.
`<root>:`	The root of this log4j.

EXAMPLE:

```
<log4j:configuration>
  <appender name="dcs" class="org.apache.log4j.FileAppender">
    <param name="File" value="logs/proxy.log" />
    <param name="Append" value="true" />
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%t %-5p %c{2} - %m%n" />
    </layout></appender>
  <category name="org.apache.log4j.xml">
    <priority value="debug" />
    <appender-ref ref="dcs" /></category>
  <root>
    <priority value="debug" />
    <appender-ref ref="dcs" /></root>
</log4j:configuration>
```

3.2 *Managed Device Proxy Configuration*

This section describes how to configure a proxy for a managed device. The device configuration file contains the information about connections, device, administration protocols information, and log4j configuration. The proxy must have the information about the connection to the server and administration protocol to connect to the device in order to build the control loop.

3.2.1 Configuration File

Configuration for a proxy is stored in XML files following the defined hierarchy.

```
<DCSProxy xmlns:log4j=":log4j="http://jakarta.apache.org/log4j/">
   <Connections>
      <GlobalParameters>
      </GlobalParameters>
      <Service>
      </Service>
      <Broadcaster>
      </Broadcaster>
   </Connections>
   <Device manufacturer="" model="" system=""\>
```

```
      <ModelVariables>
      </ModelVariables>
      <AdminProtocols>
      </AdminProtocols>
   </Device>
   <log4j:configuration>
   </log4j:configuration>
</DCSProxy>
```

3.2.1.1 Example Configuration File

This is an example of a configuration file for the proxy.

```
<DCSProxy xmlns:log4j="http://jakarta.apache.org/log4j/">
  <Connections>
    <GlobalParameters>
    </GlobalParameters>
    <Service name="control">
      <RMIChannel host="localhost" port="1099" name="RMIProxyControl"/>
    </Service>
    <Service name="command">
      <RMIChannel host="localhost" port="1099" name="RMIProxyCommand"/>
    </Service>
    <Broadcaster name="event">
     <RMIChannel host="localhost" port="1099" name="RMIServerEvent"/>
    </Broadcaster>
    <Broadcaster name="control">
     <RMIChannel host="localhost" port="1099" name="RMIServerControl"/>
    </Broadcaster>
  </Connections>

  <Device manufacturer="Cisco" model="2950T" system="IOS-12.1(22)EA4">
    <ModelVariables>
      <Value key="hostname" value="192.168.2.254"/>
      <Value key="password" value="*MotoDemo05*"/>
   </ModelVariables>
   <AdminProtocols>
     <SNMP version="2" host="192.168.2.254" port="162" debug-level="2"
      socket-type="Standard" community="public" />
     <Telnet host="192.168.2.254" port="23" user=""
       password="*MotoDemo05*" />
   </AdminProtocols>
  </Device>
  <log4j:configuration>
    <appender name="dcs" class="org.apache.log4j.FileAppender">
     <param name="File" value="logs/proxy.log" />
     <param name="Append" value="true" />
     <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern" value="%t %-5p %c{2} - %m%n" />
     </layout></appender>
    <category name="org.apache.log4j.xml">
     <priority value="debug" /><appender-ref ref="dcs" /></category>
    <root>
      <priority value="debug" /><appender-ref ref="dcs" /></root>
  </log4j:configuration>
</DCSProxy>
```

3.3 Server Configuration

This section describes how to configure the server. There must be a server configuration file in order to run the DCS server correctly.

3.3.1 Configuration File

The server configuration file contains the information about the connections, list of the file path of the system (See 3.1.3) configuration files (See 3.4.2) and device (See 3.1.2) configuration files (See 3.4.1) that the server can manage. Also, there are rule tables and log4j configuration (See 3.1.4) information in the server configuration file. The user can edit the rule table using the DCS GUI (See 4.3). The server configuration is stored in an XML file following the defined hierarchy.

```
<DCSServer xmlns:log4j="://jakarta.apache.org/log4j/">
   <Connections>
      <GlobalParameters>
      </GlobalParameters>
      <Service>
      </Service>
   </Connections>
   <Systems>
   </Systems>
   <Devices>
   </Devices>
   <MasterRuleTable>
   <RuleTable>
      <Rule/>
   </RuleTable>
   </MasterRuleTable>
   <log4j:configuration>
   </log4j:configuration>
</DCSServer>
```

3.3.2 Example Configuration File

This is an example of a configuration file for the server.

```
<DCSServer xmlns:log4j="http://jakarta.apache.org/log4j/">
<Connections>
   <GlobalParameters>
   </GlobalParameters>
    <Service name="control">
    <RMIChannel host="localhost" port="1099" name="RMIServerControl" />
    </Service>
    <Service name="event">
      <RMIChannel host="localhost" port="1099" name="RMIServerEvent" />
    </Service>
  </Connections>
  <Systems>
   <System source="conf\systems\ios-12.1.22.ea4.xml" />
   <System source="conf\systems\windows-xp-pro-sp2.xml" />
  </Systems>
  <Devices>
   <Device source="conf\devices\cisco.catalyst.2950t.xml" />
   <Device source="conf\devices\wintel-pc.xml" />
  </Devices>
  <MasterRuleTable>
   <RuleTable manufacturer="Intel" model="x86"
      system="windows-xp-pro-sp2" />
    <RuleTable manufacturer="Cisco" model="2950T"
    system="windows-xp-pro-sp2" />
    <RuleTable manufacturer="Intel" model="x86"
    system="IOS-12.1(22)EA4">
     <Rule alert="First" remedy="register" />
     <Rule alert="Second" remedy="port1ShutDown" />
     <Rule alert="Third" remedy="port1ShutDown" />
    </RuleTable>
    <RuleTable manufacturer="Cisco" model="2950T"
    system="IOS-12.1(22)EA4">
     <Rule alert="medBandwidthP1" remedy="register" />
     <Rule alert="highBandwidthP1" remedy="port1ShutDown" />
    </RuleTable>
   </MasterRuleTable>
   <log4j:configuration>
    <appender name="dcs" class="org.apache.log4j.FileAppender">
     <param name="File" value="logs/server.log" />
     <param name="Append" value="true" />
     <layout class="org.apache.log4j.PatternLayout">
       <param name="ConversionPattern" value="%t %-5p %c{2} - %m%n" />
     </layout>
    </appender>
    <category name="org.apache.log4j.xml">
     <priority value="debug" />
     <appender-ref ref="dcs" />
    </category>
    <root>
      <priority value="debug" />
      <appender-ref ref="dcs" />
    </root>
   </log4j:configuration>
</DCSServer>
```

3.4 Other Configuration Files

This section describes how to configure the Device and System configuration files. There are device configuration files and system configuration files that the user can edit.

3.4.1 Device Configuration File

The device configuration file contains information about a device name, different types of protocols available on the device and rules to define undesirable states for device. In order to manage the device, the DCS server must have the device configuration file for each device model to be managed. For the device with the SNMP Protocol, the configuration file must have the list of the path to the SNMP MIB files for the specific device model itself. Configurations for devices are stored in XML files following the defined hierarchy:

```
<Device manufacturer="" model="">
   <EventProcessors>
      <Processor type="snmp">
         <SNMPProcessor>
         </SNMPProcessor>
      </Processor>
      <Processor type="cli">
         <CLIProcessor type=""/>
      </Processor>
   </EventProcessors>

   <ModelWatcher>
      <ModelAlertEntry name=""></ModelAlertEntry>
   </ModelWatcher>
</Device>
```

Attribute values:

`manufacturer:`	The name of the device manufacturer. For example, “Cisco”.
`model:`	The name of the device model.
`EventProcessors:`	The different type of protocol exists in the device.
`Processor type:`	The type of `AdminProtocol`. Examples are "cli" or "snmp".
`ModelWatcher:`	The rules to define undesirable states for device.
`ModelAlertEntry name:`	The name of the alert entry.

3.4.1.1 Example Device Configuration File

Here is an example configuration files for Cisco Catalyst switch 2950T device.

```
<Device manufacturer="Cisco" model="2950T">
   <EventProcessors>
      <Processor type="snmp">
         <SNMPProcessor>
           <MibFile filename="mibs/cat2950/v2/TCP-MIB.my" />
           <MibFile filename="mibs/cat2950/v2/SNMP-VACM-MIB.my" />
           <MibFile filename="mibs/cat2950/v2/ENTITY-MIB.my" />
           <MibFile filename="mibs/cat2950/v2/IF-MIB.my"/>
           <MibFile filename="mibs/cat2950/v2/RMON-MIB.my"/>
           <MibFile filename="mibs/cat2950/v2/RMON2-MIB.my"/>
           <MibFile filename="mibs/cat2950/v2/TCP-MIB.my"/>
           <MibFile filename="mibs/cat2950/v2/CISCO-SYSLOG-MIB.my"/>
          </SNMPProcessor>
      </Processor>
      <Processor type="cli">
        <CLIProcessor type=" " />
      </Processor>
   </EventProcessors>
   <ModelWatcher>
      <ModelAlertEntry name="medBandwidthP1">
            <MoreOrEq a="[1.3.6.1.2.1.16.3.1.1.5.2]" b="2048" />
      </ModelAlertEntry>
      <ModelAlertEntry name="highBandwidthP1">
            <MoreOrEq a="[1.3.6.1.2.1.16.3.1.1.5.3]" b="4096" />
      </ModelAlertEntry>
   </ModelWatcher>
</Device>
```

3.4.2 System Configuration File

The system configuration file contains information about the operating system of devices. Each file contains a remedy table and command table for a operating system. The major sections are `RemedyTable`, `Remedy alert`, and `CommandTable`. The remedy table contains the list of the information about the remedies in order to put the device back to the desired state from the undesired state. The command table contains the commands, timeout value, description of the command, and dependency specific to the operating system. Configurations for a system are stored in XML files following the defined hierarchy:

```
<System type="">
  <RemedyTable>
   <Remedy name="" description="">
    <InvokeCommand name="" />
   </Remedy>
  </RemedyTable>
   <CommandTable>
      <Command name="" timeout="" async="" description="" depends="">
         [COMMAND]
      </Command>
   </CommandTable>
</System>
```

Attribute values:

`type:`	The name of the operating system.
`Remedy name:`	The name of the remedy. For example, "register". There could be multiple remedy's in a `RemedyTable`.
`description:`	The brief description of the remedy.
`InvokeCommand name:`	The name of the command to execute when the alert is received. There could be multiple `InvokeCommand` in one `RemedyAlert`.
`CommandTable:`	The command name, dependency, timeout, description and actual command.
`Command name:`	The friendly name of the command.
`timeout:`	The timeout value until the end of the command execution. This parameter is optional. Default is 5000.
`async:`	The Boolean value if the command is executed asynchronously. This parameter is optional. Default is “false”.
`description:`	The brief description of the command. This parameter is optional.
`depends:`	The name of the command that needs to be executed before this command (i.e., its dependency). This parameter is optional.
`[COMMAND]:`	The command to execute.

3.4.2.1 Example System Configuration File

Here is an example of System configuration file for IOS-12.1(22)EA4 that is an operating system of Cisco Catalyst switch 2950T device.

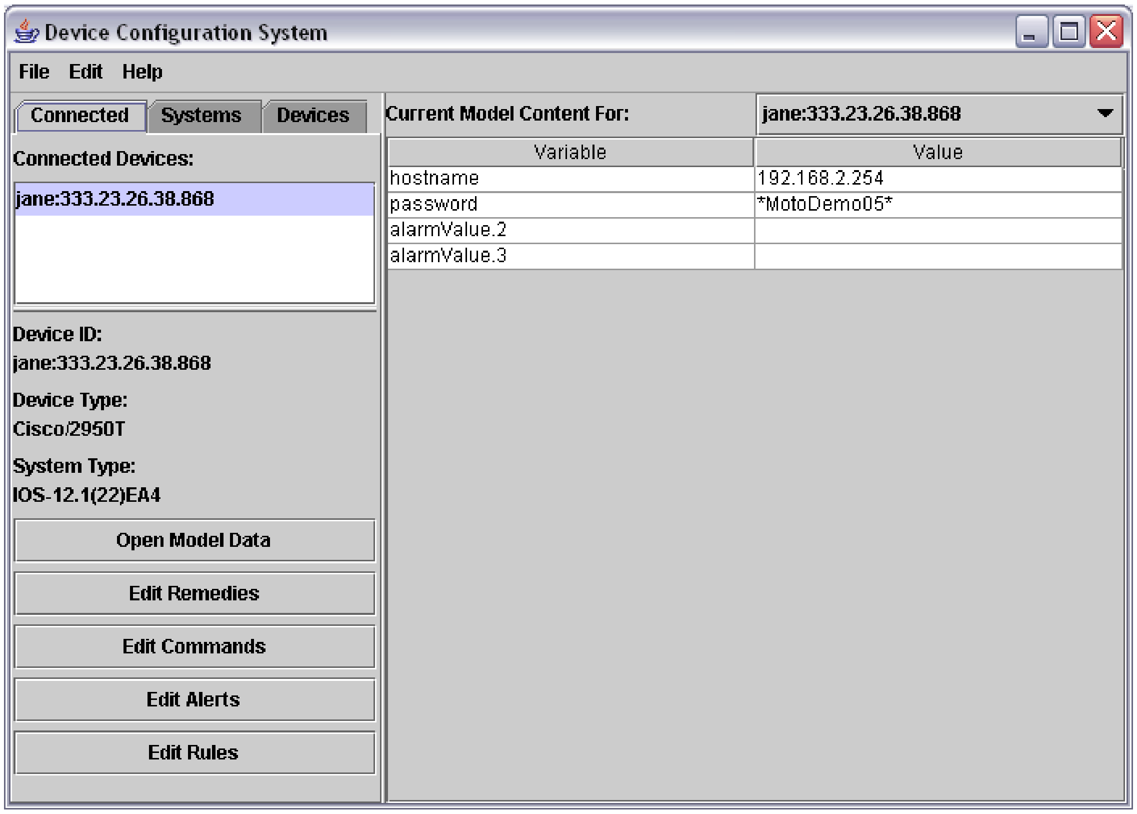
```
<System type="IOS-12.1(22)EA4">
  <RemedyTable>
    <Remedy name="register" description="Remedy invoked when a device
      running this operating system registers with DCS">
      <InvokeCommand name="ShowInterfaceStatus" />
    </Remedy>
    <Remedy name="port1ShutDown">
      <InvokeCommand name="incMaxPpsPort1" />
      <InvokeCommand name="endConfig" />
      <InvokeCommand name="enablePort1" />
      <InvokeCommand name="endConfig" />
    </Remedy>
  </RemedyTable>

  <CommandTable>
    <Command name="decMaxPpsPort1" timeout="1000"
        description="decrements the maximum number of packets per second
        for port #1" depends="interface1">
      storm-control unicast level pps 20 20
    </Command>
    <Command name="incMaxPpsPort1" timeout="5000" description="update
         storm control to a more useful packets-per-second level"
         depends="interface1">
        storm-control unicast level pps 200000 200000
    </Command>
    <Command name="enablePort1" timeout="5000" description="enables port
          #1" depends="interface1">
      no shutdown
    </Command>
    <Command name="interface1" timeout="5000" description="enters
          configuration mode for the fast ethernet interface #1"
          depends="config">interface fastethernet0/1
    </Command>
    <Command name="config" timeout="5000" description="enters
         configuration mode">
      configure terminal
    </Command>
    <Command name="enable" timeout="5000" description="sends the admin
         password to the switch" depends="enableCmnd">
      *MotoDemo05*
    </Command>
    <Command name="enableCmnd" timeout="5000" description="switches to
          priviliged mode">
      enable
    </Command>
    <Command name="endConfig" timeout="5000" description="terminates a
          configuration session">
      end
    </Command>
    <Command name="test" timeout="5000" description="test command"
          delay="15000">
      ping
    </Command>
    <Command name="ShowInterfaceStatus" timeout="5000" return-
          result="true" description="retrieves the status of interfaces
          on a cisco switch">
      show interface status
    </Command>
  </CommandTable>
</System>
```

MOTOROLA LABS

4 Graphical User Interface

The DCS Server contains the Graphical User Interface (GUI) to help the user to configure and manage the DCS control loop. The GUI resides on the server and displays when the server first executes. From the GUI, a user can modify alerts that allow DCS to understand when a problem has occurred on a managed device. A user can then create a remedy, which is a list of commands that will be executed to fix the problem. Finally, the user can use the GUI to match alerts to remedies in a rule. The following is a screen shot of the GUI for DCS.



Connected Devices:
jane:333.23.26.38.868

Device ID:
jane:333.23.26.38.868
Device Type:
Cisco/2950T
System Type:
IOS-12.1(22)EA4

Current Model Content For: jane:333.23.26.38.868

Variable	Value
hostname	192.168.2.254
password	*MotoDemo05*
alarmValue.2	
alarmValue.3	

Figure 4: Graphical User Interface of the DCS

4.1 Object Selection Panel

The object selection panel is located at left side of the GUI and contains 3 tabs. The Object Selection Panel is used to select the connected device, operating system, or device that needs to be configured.

4.1.1 Connected Tab

The connected tab is used to manage the registered devices that are currently connected to the server. Detailed information of each device is shown by clicking an entry on the Connected Devices list. The Connected Devices list is automatically updated every time a device is registered and un-registered. The buttons are enabled by selecting a device in the Connected Devices list.

The Open Model Data button opens the Model Content Viewer (see section 4.2.1) which displays the current model contents for the selected device.

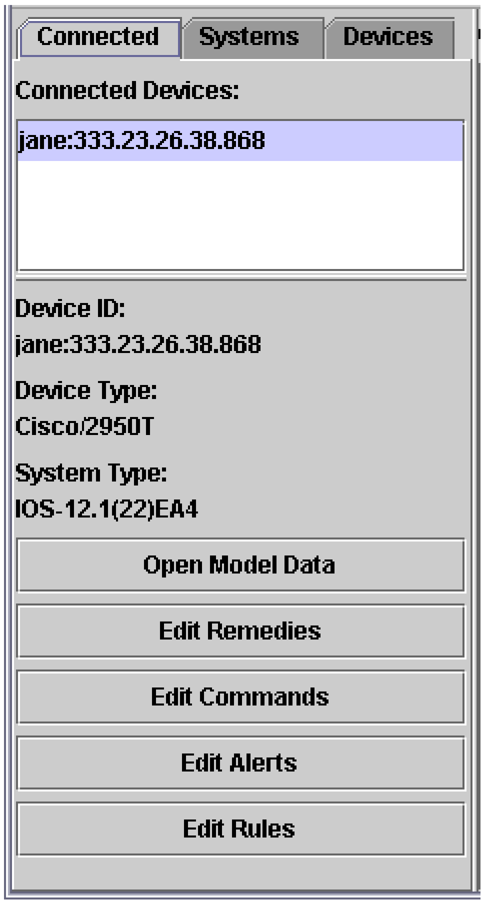
The Edit Remedies button displays the remedies for the operating system of the selected device (see section 2.2.2).

The user can edit remedies for the selected device on the remedy setting (see section 4.3).

The Edit Commands button displays the commands for the operating system of the selected device (see section 4.3).

The Edit Alerts button opens device setting for the selected device (see section 4.3).

The Edit Rule button opens the Rule Table for the selected device (see section 4.3).



Connected Devices:
jane:333.23.26.38.868

Device ID:
jane:333.23.26.38.868

Device Type:
Cisco/2950T

System Type:
IOS-12.1(22)EA4

Figure 5: Connected Tab

4.1.2 Systems Tab

The Systems Tab is used to manage the remedies, commands, and the rules for the operating system. The Configured System listbox displays a list of the operating systems that are configurable by DCS. Also, there are four buttons which are Create New System, Edit Remedies, Edit Commands, and Edit Rules Config.

The Create New System button allows the user to add the new system on the Configured System listbox.

The Edit Remedies button allows the user to edit remedies for the selected operating system (See section 4.3).

The Edit Commands button allows the user to edit commands for the selected operating system (See section 4.3).

The Edit Rules Config button allows the user to edit rules for the selected operating system (See section 4.3)

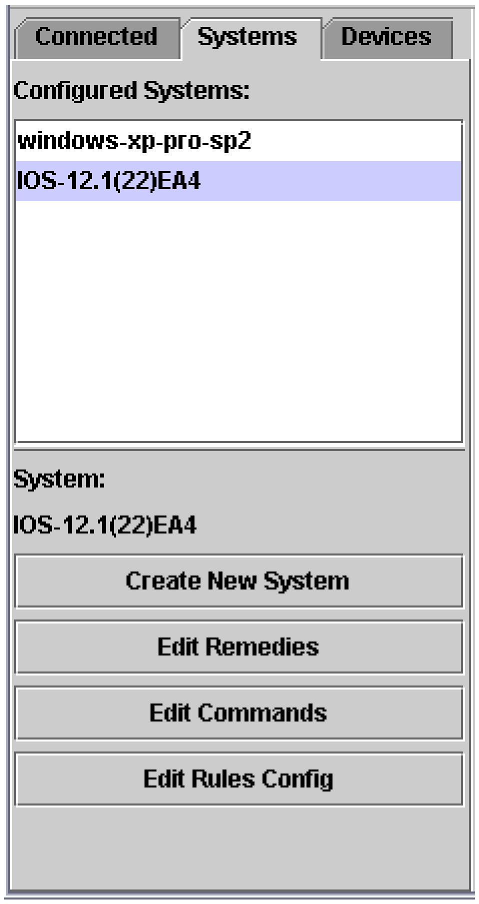


Figure 6: Systems Tab

4.1.3 Devices Tab

The Devices Tab is used to manage the alerts and rules of the devices. The Configured Device Types listbox displays a list of the devices that are configurable by DCS. Also, there are three buttons which are Create New Device, Edit Alerts, and Edit Rules.

The Create New Device button allows the user to add the new device on the Configured Device Types listbox.

The Edit Alerts button allows the user to edit alerts for the selected device (See section 4.3).

The Edit Rules button allows the user to edit the rules for the selected device (See section 4.3)

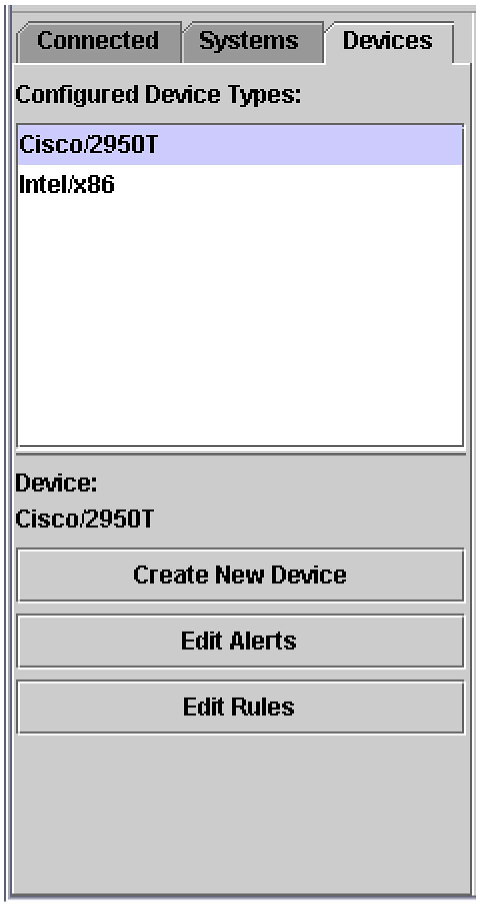


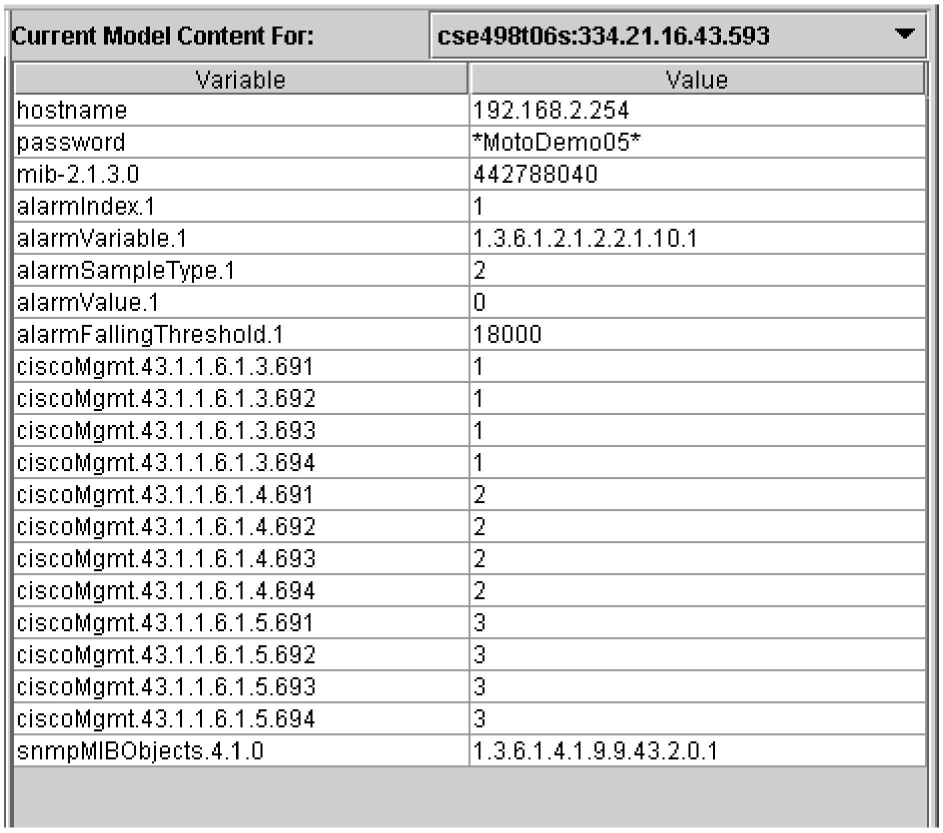
Figure 7: Devices Tab

4.2 Configuration Panel

The Configuration Panel is located at the right side of the GUI. The user can obtain and modify the information about the selected system or device on the Configuration Panel. Also, the user can see the current state of the connected device.

4.2.1 Model Content Viewer

The Model Content Viewer displays the current state of a device which is connected (registered) to the server. The model content contains the variables of the managed device that are monitored by the server. As the values of the variable changes, the state of the device may be changed to the desired or undesired state. The Model Content Viewer can be accessed from the Object Selection Panel by selecting the Connection tab, selecting a connected device, then pressing Open Model Data button. The user can also obtain the model content of connected devices by selecting from a pull down menu on the top of the Configuration Panel. The name of the Variable is not editable, but the value that corresponds to each Variable can be changed by double clicking the cell and typing in the value.

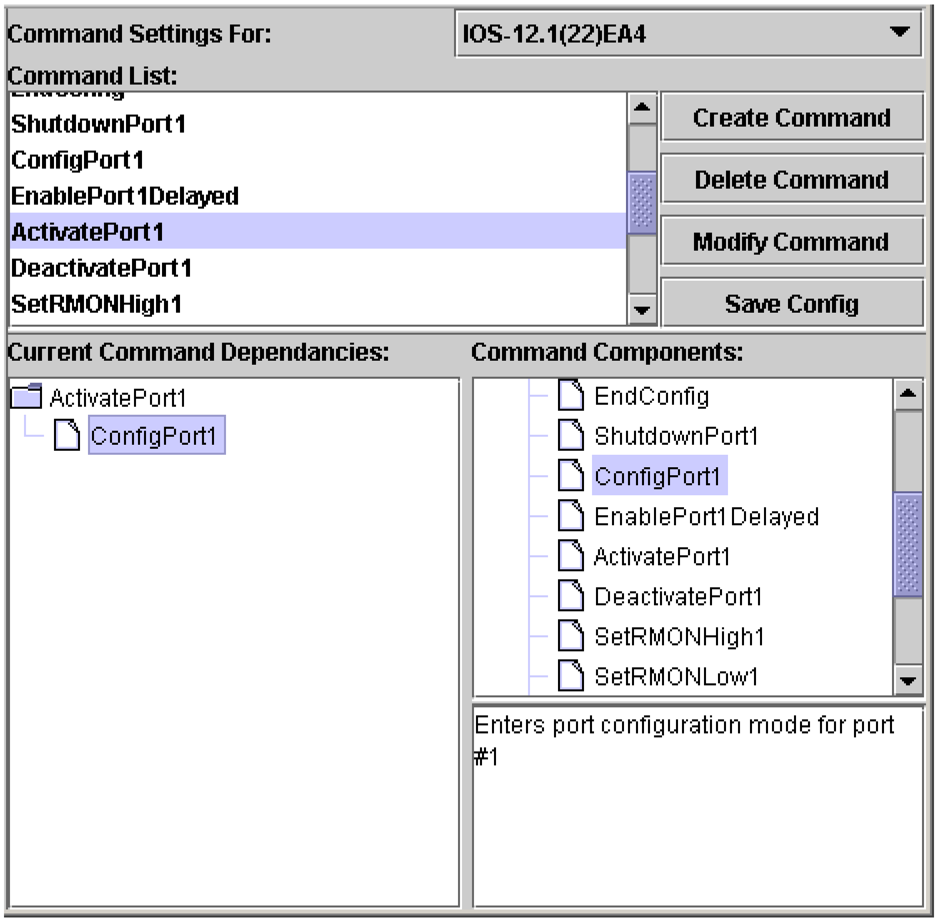


Variable	Value
hostname	192.168.2.254
password	*MotoDemo05*
mib-2.1.3.0	442788040
alarmIndex.1	1
alarmVariable.1	1.3.6.1.2.1.2.2.1.10.1
alarmSampleType.1	2
alarmValue.1	0
alarmFallingThreshold.1	18000
ciscoMgmt.43.1.1.6.1.3.691	1
ciscoMgmt.43.1.1.6.1.3.692	1
ciscoMgmt.43.1.1.6.1.3.693	1
ciscoMgmt.43.1.1.6.1.3.694	1
ciscoMgmt.43.1.1.6.1.4.691	2
ciscoMgmt.43.1.1.6.1.4.692	2
ciscoMgmt.43.1.1.6.1.4.693	2
ciscoMgmt.43.1.1.6.1.4.694	2
ciscoMgmt.43.1.1.6.1.5.691	3
ciscoMgmt.43.1.1.6.1.5.692	3
ciscoMgmt.43.1.1.6.1.5.693	3
ciscoMgmt.43.1.1.6.1.5.694	3
snmpMIBObjects.4.1.0	1.3.6.1.4.1.9.9.43.2.0.1

Figure 8: Model Content Viewer

4.2.2 System Configuration

The System Configuration modifies the configuration file of each operating system that the server supports. The System Configuration is used to edit the commands available for each operating system. The user can get to the system configuration by clicking the Systems tab and pressing the “Edit Commands” button.

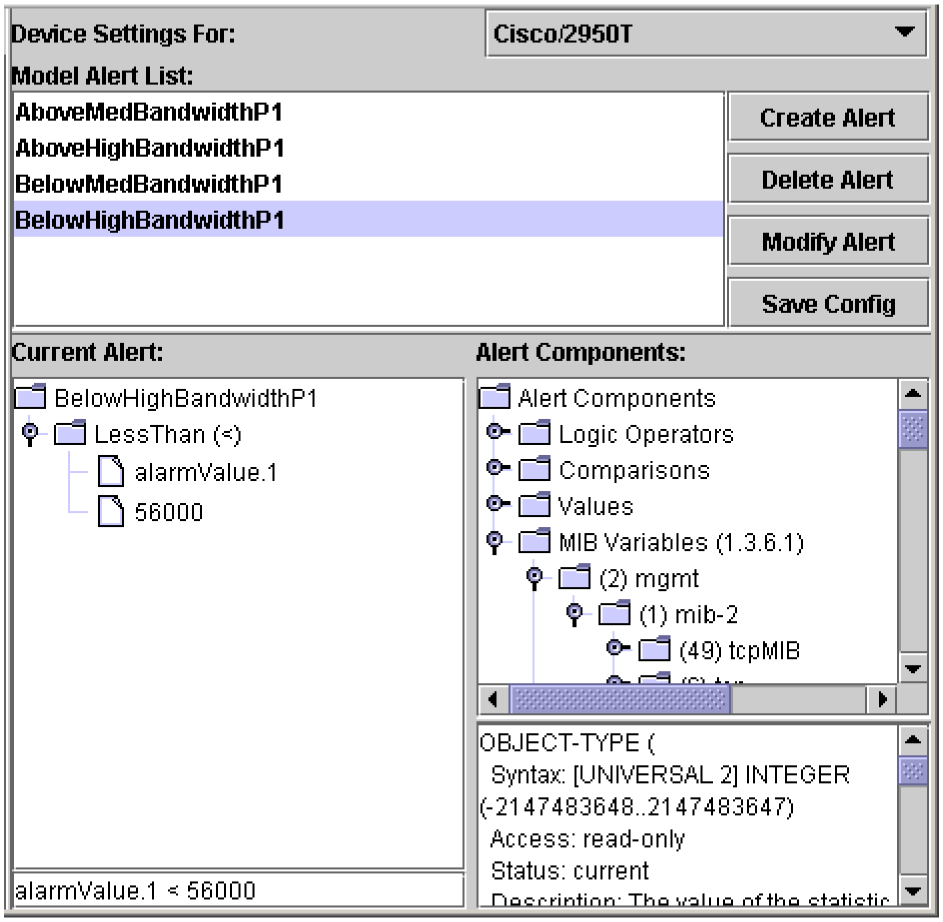


Enters port configuration mode for port #1

Figure 9: System Configuration

4.2.3 Device Configuration

The Device Configuration is used to edit the alerts settings. The user can get to the device configuration by clicking the “Devices” tab and clicking “Edif Alert” button. (See 4.3).



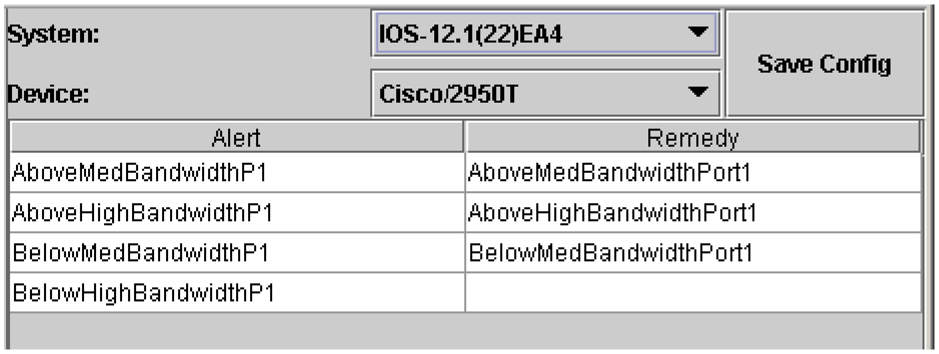
alarmValue.1 < 56000

OBJECT-TYPE (
Syntax: [UNIVERSAL 2] INTEGER (-2147483648..2147483647)
Access: read-only
Status: current

Figure 10: Device Configuration

4.2.4 Rule Table Configuration

The rule table configuration is for defining the solution to change the state of device from the undesired state to desired state. The Rule Table Configuration is used to create a rule by associating an alert with a remedy. The User can get to the rule table configuration by pressing “Edit Rules” button.



System: IOS-12.1(22)EA4

Device: Cisco/2950T

Alert	Remedy
AboveMedBandwidthP1	AboveMedBandwidthPort1
AboveHighBandwidthP1	AboveHighBandwidthPort1
BelowMedBandwidthP1	BelowMedBandwidthPort1
BelowHighBandwidthP1	

Figure 11: Rule Table Configuration setting of GUI

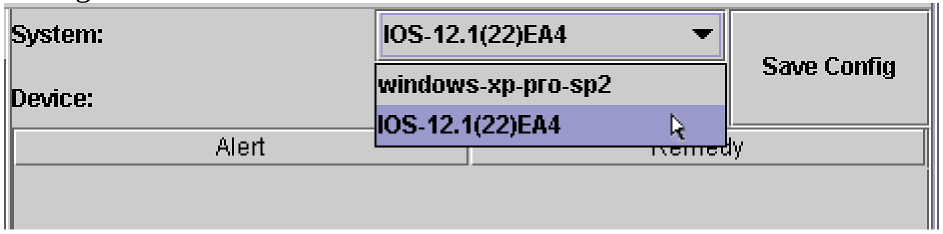
4.3 Common GUI Tasks

The common GUI tasks include creating a rule, deleting a rule, modifying an exiting rule, creating a command, deleting a command, modifying an existing command, creating a remedy, deleting a remedy, modifying an exiting remedy, creating an alert, deleting an alert, and modifying an exiting alert. After editing rules, commands, remedies, or alerts, the user can save the settings to restart the server and proxy with the edited configuration.

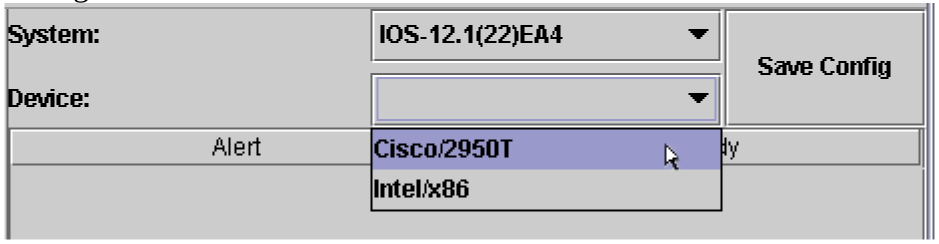
4.3.1 Creating a rule

<u>Follow these steps to create a rule:</u>

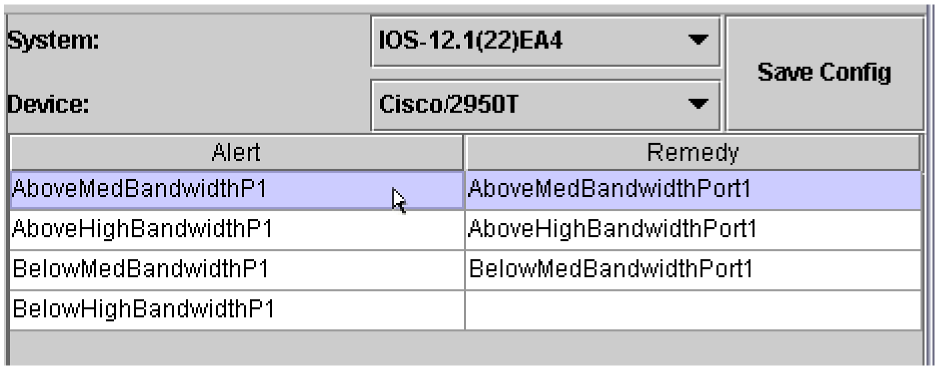
1) Create an alert for the device (See section 4.3.10)
2) Create an remedy for the system (See section 4.3.7)
3) Press "Edit Rules" button on the Object Selection Panel to go to Rule Table Configuration.
4) Select the system from the pulldown menu which is at the top of the Configuration Panel.



5) Select the device from the pulldown menu which is at the top of the Configuration Panel.

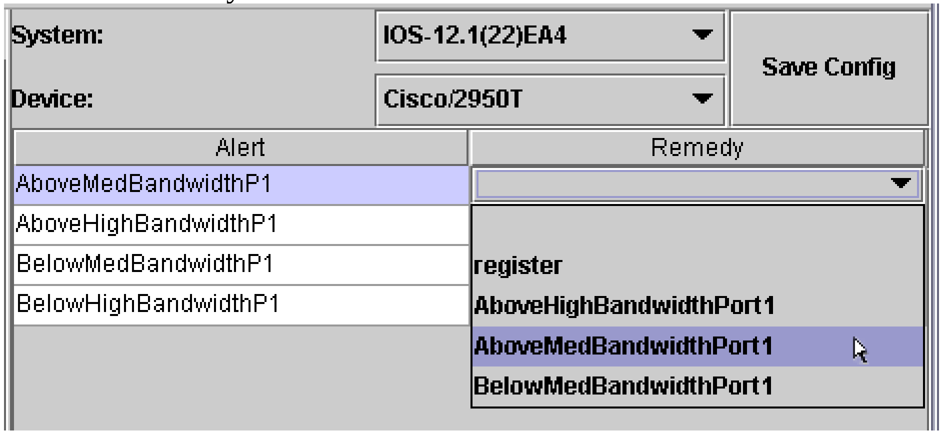


6) Select the Alert.



Alert	Remedy
AboveMedBandwidthP1	AboveMedBandwidthPort1
AboveHighBandwidthP1	AboveHighBandwidthPort1
BelowMedBandwidthP1	BelowMedBandwidthPort1
BelowHighBandwidthP1	

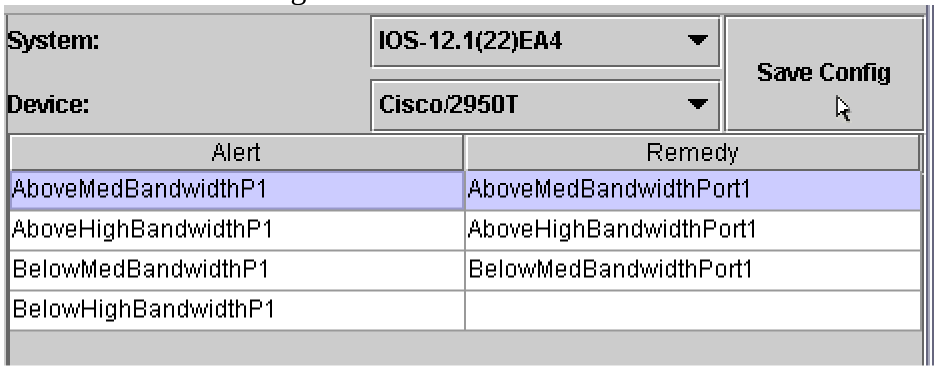
7) Select the Remedy to use for the Alert.



System: IOS-12.1(22)EA4
Device: Cisco/2950T

Alert	Remedy
AboveMedBandwidthP1	
AboveHighBandwidthP1	
BelowMedBandwidthP1	
BelowHighBandwidthP1	

8) Select the “Save Config” button.



System: IOS-12.1(22)EA4
Device: Cisco/2950T

Alert	Remedy
AboveMedBandwidthP1	AboveMedBandwidthPort1
AboveHighBandwidthP1	AboveHighBandwidthPort1
BelowMedBandwidthP1	BelowMedBandwidthPort1
BelowHighBandwidthP1	

4.3.2 Deleting a rule

Follow these steps to delete a rule:

1) Select the system from the pulldown menu which is at the top of the Configuration Panel.
2) Select the device from the pulldown menu which is at the top of the Configuration Panel.
3) Select the empty (“”) entry for the remedy.
4) To save the current rule configuration, click the “Save Config” button.

4.3.3 Modifying an existing rule

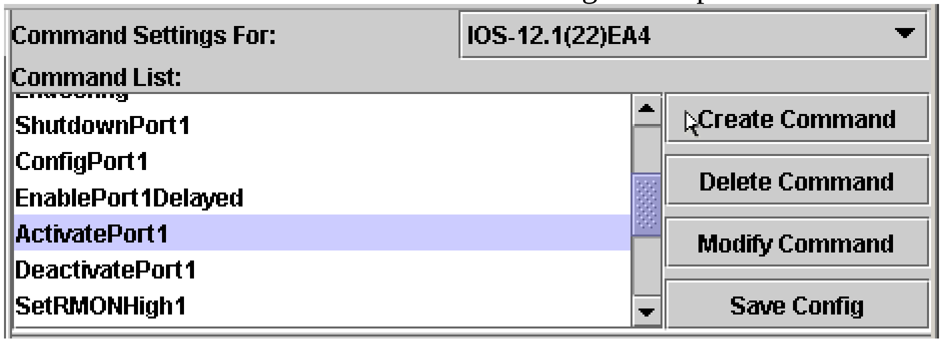
Follow these steps to modify an existing rule:

1) Select the system from the pulldown menu which is at the top of the Configuration Panel.
2) Select the device from the pulldown menu which is at the top of the Configuration Panel.
3) Select the Alert.
4) Select the Remedy to use for the Alert.
5) To save the current rule configuration, click the “Save Config” button.

4.3.4 Creating a command

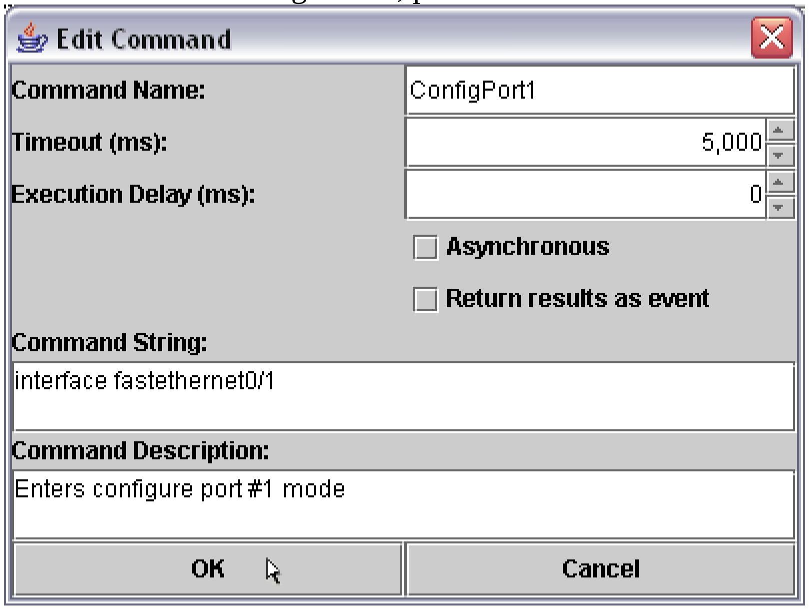
<u>Follow these steps to create a command:</u>

1) Press "Edit Commands" button on the object selection panel.
2) Press "Create Command" button on the configuration panel.



ShutdownPort1
ConfigPort1
EnablePort1Delayed
ActivatePort1
DeactivatePort1
SetRMONHigh1

3) Type the new command name, command string. The timeout, execution delay, asynchronous, return result as event, and command description values are optional. After done entering values, press "OK" button.

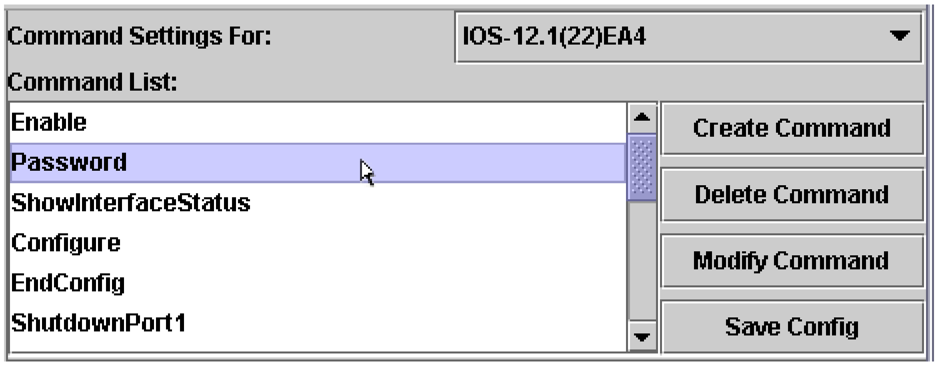


Command Name: ConfigPort1
Timeout (ms): 5,000
Execution Delay (ms): 0
☐ Asynchronous
☐ Return results as event
Command String:
interface fastethernet0/1
Command Description:
Enters configure port #1 mode

4.3.5 Deleting a command

Follow these steps to delete a command:

1) Press "Edit Commands" button on the object selection panel.
2) Select the name of the command that needs to be deleted from the Command List.

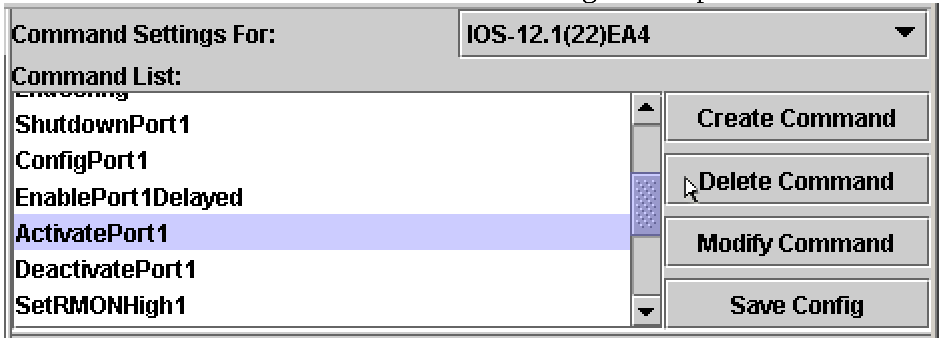


Command Settings For: IOS-12.1(22)EA4

Command List:

Enable
Password
ShowInterfaceStatus
Configure
EndConfig
ShutdownPort1

3) Press "Delete Command" button on the configuration panel.



Command Settings For: IOS-12.1(22)EA4

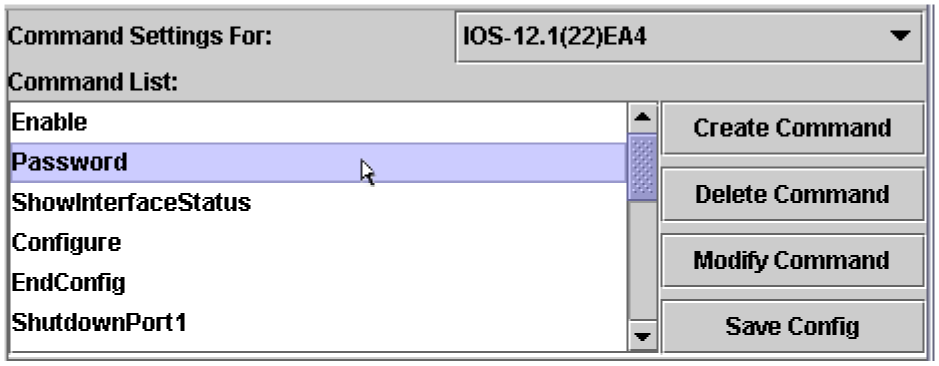
Command List:

ShutdownPort1
ConfigPort1
EnablePort1Delayed
ActivatePort1
DeactivatePort1
SetRMONHigh1

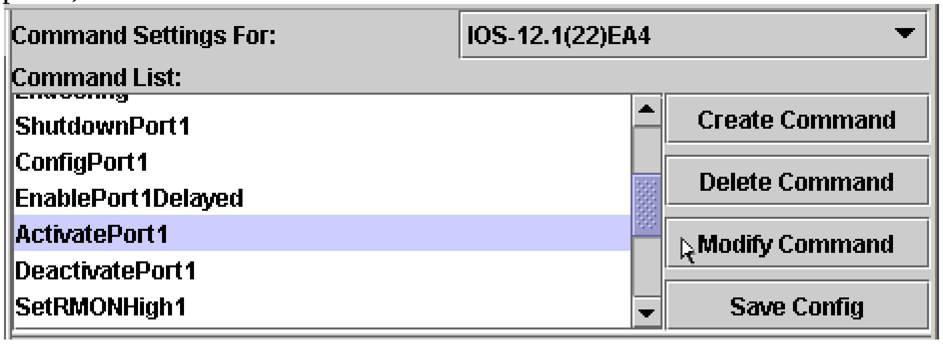
4.3.6 Modifying an existing command

Follow these steps to modify an existing command:

1) Select the name of the command that needs to be edited from the Command List.



2) Expand Command Components to select the components that needs to be added to the Current Command Dependencies.
3) To edit the commands, press the “Modify Command” button on the configuration panel, and edit the values.

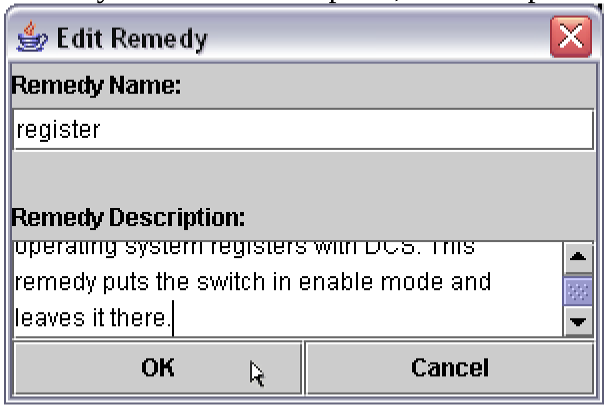


4) To add the commands to the current command dependencies, drag and drop the selected component from the Command Components to the Current Command Dependencies.
5) To remove the commands from the current command dependencies, drag and drop out the selected component from the Current Command Dependencies.
6) To save the edited commands, press “Save Config” button.
7) To configure the comments on the different system, select the other system from the Configured System listbox on the Object Selection Panel or pulldown menu on the top of the Configuration Panel.

4.3.7 Creating a remedy

Follow these steps to create a remedy:

1) Press “Edit Remedies” button on the object selection panel.
2) Press “Create Remedy” button on the configuration panel.
3) Type the new remedy name and description, and then press “OK”.



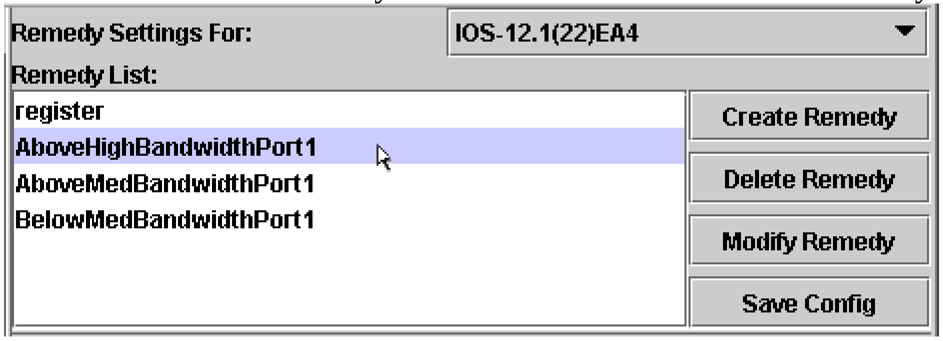
Remedy Name:
register

Remedy Description:
operating system registers with DCS. This
remedy puts the switch in enable mode and
leaves it there.

4.3.8 Deleting a remedy

Follow these steps to create a remedy:

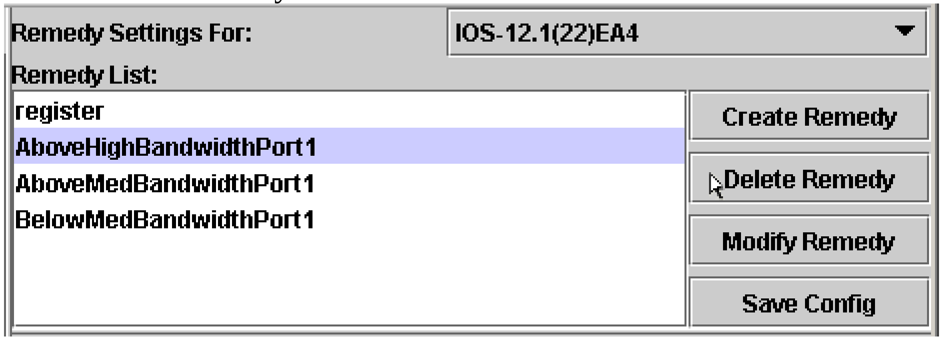
1) Press “Edit Remedies” button on the object selection panel.
2) Select the name of the remedy that needs to be deleted from the Remedy List.



Remedy Settings For: IOS-12.1(22)EA4

Remedy List:
register
AboveHighBandwidthPort1
AboveMedBandwidthPort1
BelowMedBandwidthPort1

3) Press “Delete Remedy” button.



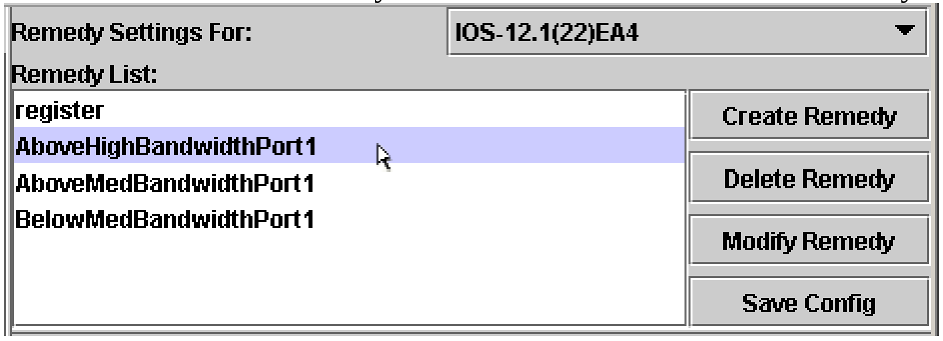
Remedy Settings For: IOS-12.1(22)EA4

Remedy List:
register
AboveHighBandwidthPort1
AboveMedBandwidthPort1
BelowMedBandwidthPort1

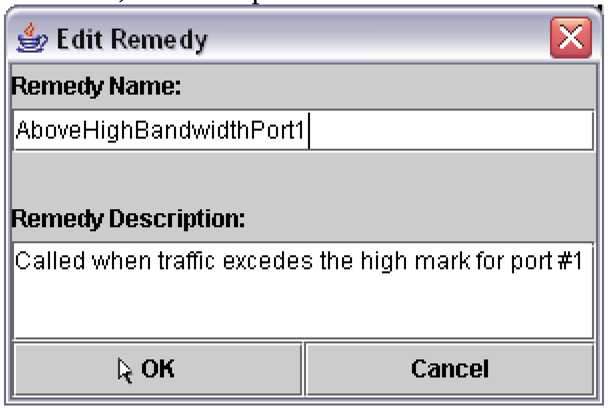
4.3.9 Modifying an Existing remedy

Follow these steps to modify an existing remedy:

1) Press "Edit Remedies" button on the object selection panel.
2) Select the name of the remedy that needs to be edited from the Remedy List.



3) To edit the remedy name or remedy decription, press "Modify Remedy" button and modify the values, and then press "OK".



Remedy Name: AboveHighBandwidthPort1

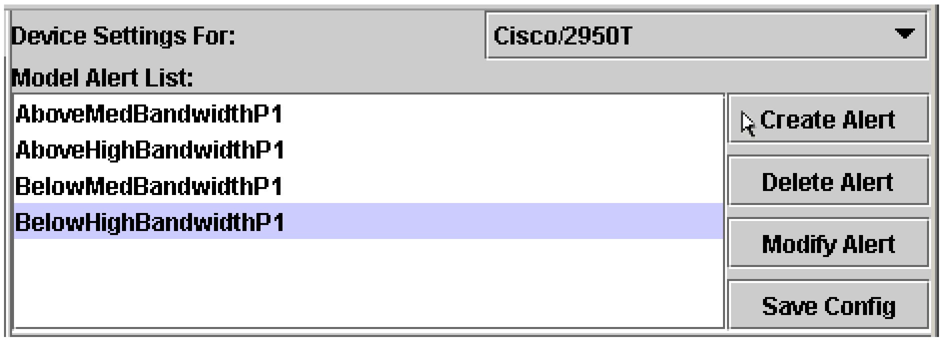
Remedy Description: Called when traffic excedes the high mark for port #1

4) To add the Remedy Components to the current remedy, expand Remedy Components to select the components that needs to be added to the Current Remedy, and drag and drop the selected component from the Remedy Components to the Current Remedy.
5) To remove the Remedy Components from the current remedy, select the Remedy Components on the Current Remedy, and drag and drop out the Remedy Components.
6) To save the edited remedies, press "Save Config" button.
7) To configure the different system, select the other system from the Configured System listbox on the Object Selection Panel or pulldown menu on the top of the Configuration Panel.

4.3.10 Creating an alert

<u>Follow these steps to create an alert:</u>

1) Press the “Edit Alerts” button on the object selection panel.
2) Press the “Create Alert” button.



Device Settings For: Cisco/2950T

Model Alert List:

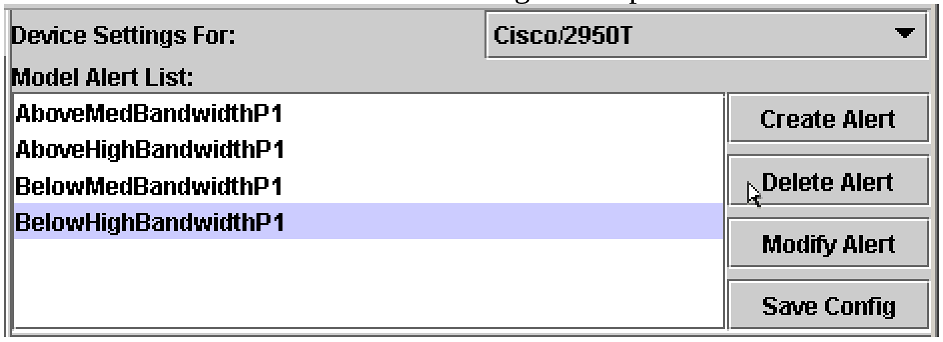
AboveMedBandwidthP1
AboveHighBandwidthP1
BelowMedBandwidthP1
BelowHighBandwidthP1

3) Type the new alert name and press ok.

4.3.11 Deleting an alert

<u>Follow these steps to delete an alert:</u>

1) Press the “Edit Alerts” button on the object selection panel.
2) Select the name of the alert that needs to be deleted from the Model Alert List.
3) Press “Delete Alert” button on the configuration panel.



Device Settings For: Cisco/2950T

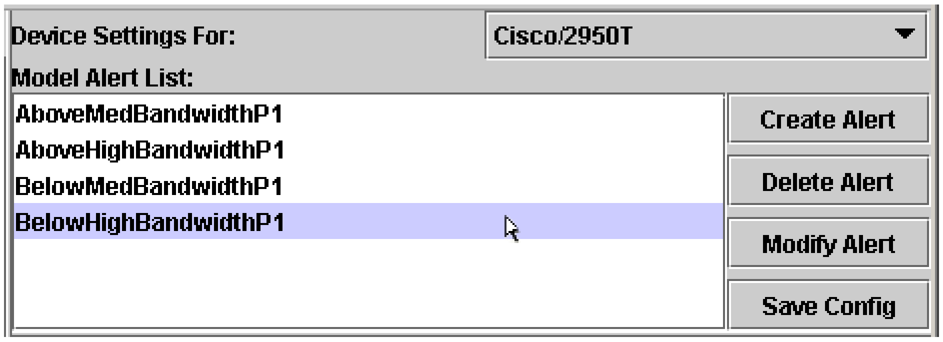
Model Alert List:

AboveMedBandwidthP1
AboveHighBandwidthP1
BelowMedBandwidthP1
BelowHighBandwidthP1

4.3.12 Modifying an existing alert

Follow these steps to modify an existing alert:

1) Press the "Edit Alerts" button on the object selection panel.
2) Select the name of the alert that needs to be edited from the Model Alert List.



3) To edit the Alert name, press "Modify Alert" button on the configuration panel and enter the new alert name.
4) To add the Alert Components to the current alert, expand Alert Components to select the components that needs to be added to the Current Alert, and drag and drop the selected component from the Alert Components to the Current Alert.
5) If the selected Alert Component was a constant value, the user must enter a value.
6) If the selected Alert Component was a MIB variable and it was from the MIB table, then the user must enter the column number.
7) To remove the Alert Components from the current alert, select the Alert Components on the current alert, and drag and drop out the Alert Components.
8) To save the edited alerts, press "Save Config" button.
9) To configure the different device, select the other device from the Configured Device Types listbox on the Object Selection Panel or pulldown menu on the top of the Configuration Panel.

5 Execution

In this section you will find the necessary steps to execute DCS.

5.1 Pre-Requisites

DCS runs with version 1.4.2 of the Java Runtime Environment (JRE). You can get the JRE at ③. You must also have the following libraries in present in the lib sub-directory:

commons-net-1.4.0.jar ⑤

grammatical-bin-1.4.jar ⑥

mibble-mibs-2.5.jar ⑥

mibble-parser-2.5.jar ⑥

jdom.jar ⑦

jsch-0.1.21.jar ⑧

log4j-1.2.8.jar ⑨

snmp4_13.jar ⑩

For information about these libraries see ④.

5.2 Command Line Arguments

The proxy requires a '-c' argument that will list the location of the proxy's configuration file (i.e., edu.msu.dcs.proxy.ManagedDeviceProxy -c proxy.config).

The Configuration Server requires a '-c' argument that lists the location of the server's configuration file (i.e., edu.msu.dcs.server.ConfigurationServer -c config.file).

5.3 Web Services Support

To support web service communication between a proxy and the server you must have Systinet WASP installed, you can find information about this product at ⑪. Currently, the Systinet server libraries and configuration files are put into a directory within the DCS distribution called "wasp". If this sub-directory is present, no further setup (other than that required by the DCS server and proxy configuration files as described in section 3.2 and 3.3) should be needed in order to use SOAP web services for communication between a proxy and DCS.

5.4 Java RMI Support

To use Java RMI for communication between the server and a proxy, an RMI registry must be available on both machines. The RMI registry will manage the list of objects which are available for a remote host to use. The Java RMI registry program (rmiregistry.exe on a Windows machine, or just rmiregistry on Linux/Unix) can be found in the bin directory within your Java Runtime Environment installation.

5.4.1 Execute Proxy and Server on the Same Machine from the Command Line

Follow these steps to execute the proxy and server on the same machine from the command line:

1) Change directories into the dist folder
2) Run the command "start-rmi.cmd"
3) Run "start server.cmd"
4) Run "start proxy.cmd"

5.4.2 Execute Proxy and Sever on Different Machines from the Command Line

To run the proxy and server on different machines, you need to modify the proxy configuration file and server.cmd. Any "Broadcaster" element in the proxy configuration needs to be changed to point to the location of the server's rmi registry (See 3.1.1.4). Likewise, the server.cmd must be updated to point to the server's codebase.

<u>After updating the proxy configuration file, do the following for the server:</u>

1) Open server.cmd and replace "localhost" in the first SET command with the host-name of the machine which will run the server. Note that this host name must be visible to all of the proxies which will connect to the server (i.e., localhost will not work for proxies that are not on the same machine as the server).
2) On the server, run the command "start server.cmd"

<u>Example of server.cmd when the server's ip address is 35.9.22.205 and the server's codebase is located on port 2001:</u>

```
@echo off

REM directory strings
   set SERV_BASE=http://35.9.22.205:2001/server.jar

REM setup server definitions
   set SERV_DEFS=-Djava.security.policy=policy.all
   set SERV_DEFS=%SERV_DEFS% -Djava.rmi.server.codebase=%SERV_BASE%

REM start the server
   java %SERV_DEFS% -jar server.jar -c conf/server.xml
```

After starting the server, do the following for the proxy:

1) Open proxy.cmd and replace "localhost" in the first SET command with the host-name of the machine which will run the proxy. Note that this name must be visible to the machine which is running DCS. (i.e., localhost will not work for servers that are not on the same machine as the proxies).
2) On the machine that will host the proxy, run "start proxy.cmd"

Example of proxy.cmd when the proxy address is 35.9.26.64:

```
@echo off

REM directory strings
   set PROX_BASE=http://35.9.26.64:2001/proxy.jar

REM setup proxy definitions
   set PROX_DEFS=-Djava.security.policy=policy.all
   set PROX_DEFS=%PROX_DEFS% -Djava.rmi.server.codebase=%PROX_BASE%

REM start the device proxy
   java %PROX_DEFS% -jar proxy.jar -c conf/proxy.xml
```

6 References

① http://snmp.westhawk.co.uk/

② http://logging.apache.org/log4j/docs/documentation.html

③ http://java.sun.com/j2se/1.4.2/download.html

④ Technical Specifications for the Device Configuration System.

⑤ http://jakarta.apache.org/commons/net/download.html

⑥ http://www.mibble.org/download/index.html

⑦ http://www.jdom.org/dist/binary/

⑧ http://www.jcraft.com/jsch/index.html

⑨ http://logging.apache.org/site/binindex.cgi

⑩ http://snmp.westhawk.co.uk/index.html

⑪ http://www.systinet.com/products/systinet_server