



CISCO IOS XML RULE EDITOR USER GUIDE

Version 1.0

Table of Contents

1. INTRODUCTION.....	3
1.1 Getting Started.....	3
1.2 User Interface Layout.....	4
1.3 Workspace.....	5
1.4 Retrieving Command Output.....	5
2. Rule Editor Workflow.....	6
2.1 Copy Command Output	6
2.2 Develop Data Model.....	7
2.3 Generate Specification File.....	9
3. Defining Table Rules.....	10
3.1 Table Command Output.....	10
3.2 Building A Table Model	10

1. INTRODUCTION

Cisco IOS devices can process the output of exec mode commands executed on the command line interface into XML. Generating XML from command output is achieved by post-processing the command output based on the rules defined in an XML file called a Rule specification file. Using the definitions in a Rule file, a Cisco IOS device scans the command output, extracts all relevant text data, and returns XML tagged output based on the tag names defined in the Rule file.

Developing Rule files for a given command can mean writing text based rules by hand. Writing a Rule specification file by hand can not only be tedious, but increases the possibility of errors that can produce incorrect XML output.

The Rule Editor application was made to aid developers with the various steps involved in drafting, validating, and deploying Rule specification files. In particular, Rule Editor helps developers:

- Define an operational data model from command output using a GUI interface
- Generate Rule specification files based on the given data model
- Generate XSD files that clients can use with Rule files
- Test Rule files with command output to check what XML output would be produced

There are generally two types of rules that appear often in Rule specification files: single property or scalar rules and table rules. A scalar rule consists of a one-to-one mapping between a property and a value while a table rule consists of a one-to-many mapping between a property and multiple values. Scalar rules are generally used to extract specific pieces of data from command output and table rules are geared towards extracting whole segments of data. Section 2 Rule Editor Workflow gives a walkthrough of how to set up scalar rules and Section 3 Defining Table Rules gives more details on how to define table rules.

1.1 Getting Started

The Rule Editor application is a single executable Java Jar file that can be downloaded from the Cisco CDN website at <http://developer.cisco.com/web/xmlmi/home>. Rule Editor is capable of running on Windows, Macintosh, and Linux, provided the host system has installed the Java Runtime Environment (JRE) prior to starting the Rule Editor application. To summarize:

1. Download and install the JRE (this guide recommends using version 1.6.0_22 or higher)
2. Double click the Rule Editor Jar file to start the application

1.2 User Interface Layout

This section of the guide gives a more in-depth look into the layout and functionality of Rule Editor's user interface. The layout is divided into four main sections:

A) Menu bar – This menu contains File, Model, Projects, and Help.

B) Projects Explorer – This view displays a hierarchical representation of the current projects in a workspace.

C) Main panel – This is the panel where most of the work will be done. This panel contains the CLI Output text area where command output to be modeled is copied, and includes four tabs that display different views of the project including the command output, the generated specification, the XSD for the specification file, and the XML test output.

D) Console/Errors panel – This panel displays messages and any errors that might occur when Rule Editor is processing data (see figure 2.1.1 for more details).

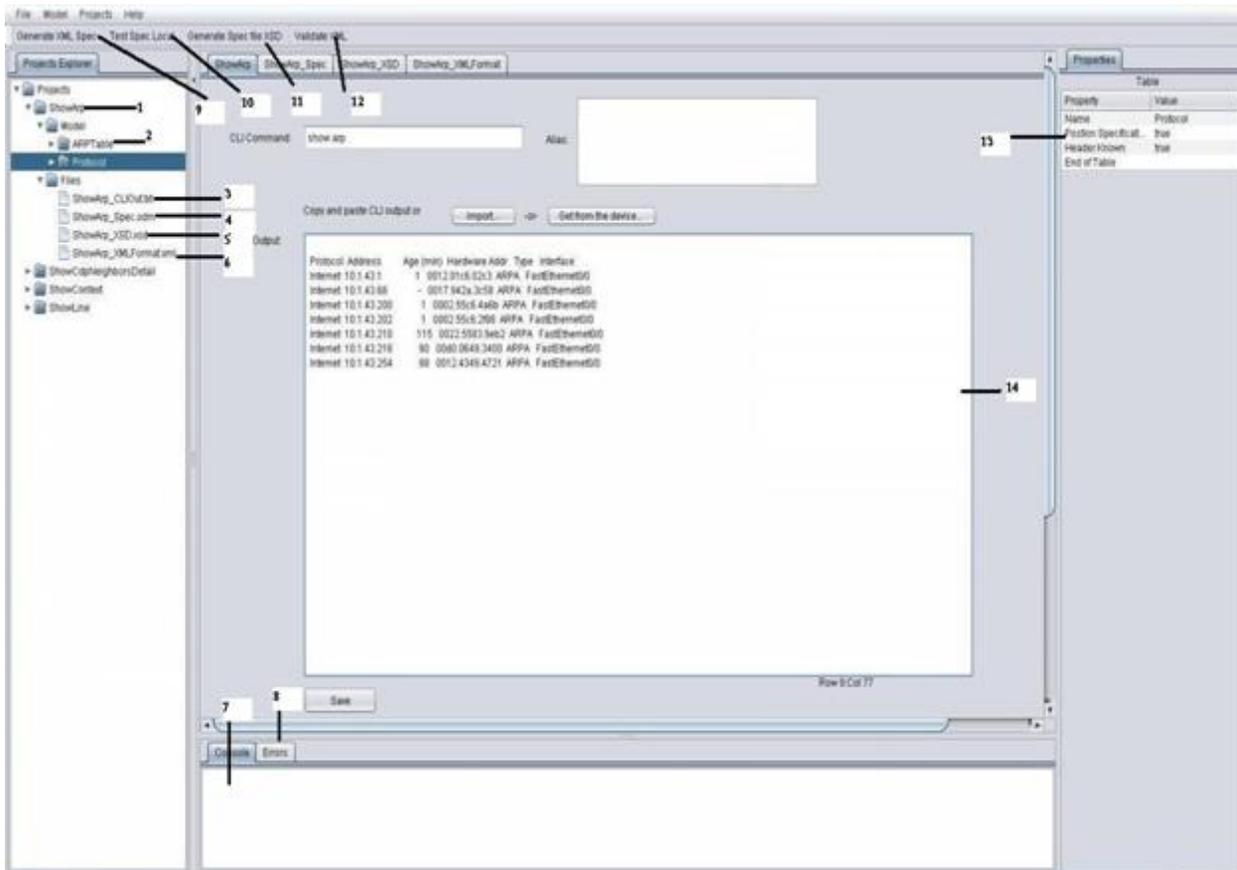


Figure 2.1.1

1 = Command used to create the specification file

2 = Collection of properties that are used to generate specification files

- 3 = Text output file
- 4 = Rule Editor specification file
- 5 = XSD schema of the specification file
- 6 = Test results of the XML output made with the Rule file
- 7 = Console log area
- 8 = Error log area
- 9 = Generates a spec file from the properties specified in the model
- 10 = Using the current specification file, transform the CLI text output into XML
- 11 = Generate the XSD file for the specification file
- 12 = Validates the XSD file against the CLI xml output
- 13 = Shows the property attributes of the currently highlighted rule
- 14 = The CLI Output text area that contains the original command output

1.3 Workspace

A workspace in Rule Editor is the directory where all files for a Rule Editor project are stored. These files include Rule specification files, XSD files, CLI command output files, and configuration files used by Rule Editor to store details on existing projects. Whenever a project is created in Rule Editor, that project is saved to the current workspace. You can switch the current workspace by selecting the Switch Workspace option from the File menu (see figure 2.2.1). Refer to section 2 Rule Editor Work flow for information on how to create a new project.

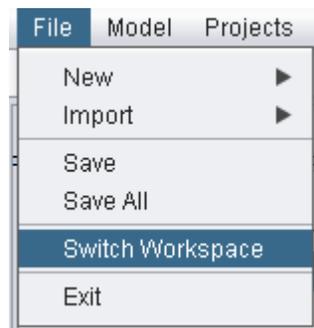


Figure 2.2.1

1.4 Retrieving Command Output

Command output can be copied and pasted into the CLI Output text area, but there are two other ways output can be copied into Rule Editor. The first way is to import the command output from a disk drive using the import functionality. To import data, click on the import... button (see figure 2.3.1), browse to the directory that contains the command output, and open the file containing the output. Rule Editor will copy the text data from the file into the CLI Output text area.

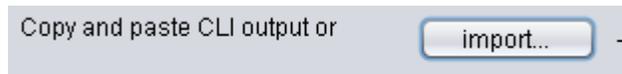


Figure 2.3.1

The second way to obtain command output is to retrieve the output directly from the Cisco IOS device. To obtain the output directly from a Cisco IOS device, click the Get from the device... button (see figure 2.3.2), supply the Host name, Username, and Password information to use when connecting to the system, and click the OK button. Be aware that this method of output retrieval requires WSMA to be configured on the Cisco IOS device. For information on how to configure WSMA, refer to the WSMA user guide.

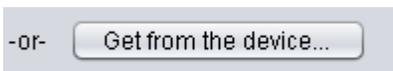


Figure 2.3.2

2. Rule Editor Workflow

To generate a Rule specification file, three steps are necessary:

1. Copy the command output into the Rule Editor application
2. Develop a data model based on the command output
3. Generate a specification file based on the data model

2.1 Copy Command Output

Create a new project in Rule Editor (see figure 1.1.1 below).

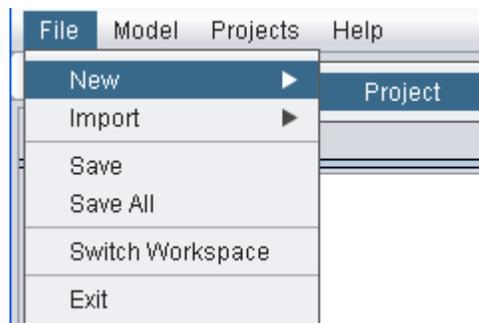


Figure 1.1.1

After clicking on the Project option, type the show command to model into the Create new project dialog box (see figure 1.1.2).

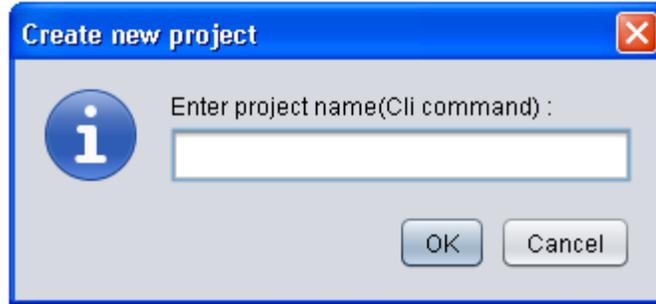


Figure 1.1.2

When Rule Editor finishes creating the new workspace, copy the show output into the the CLI Output text area (see figure 1.1.3 for example).

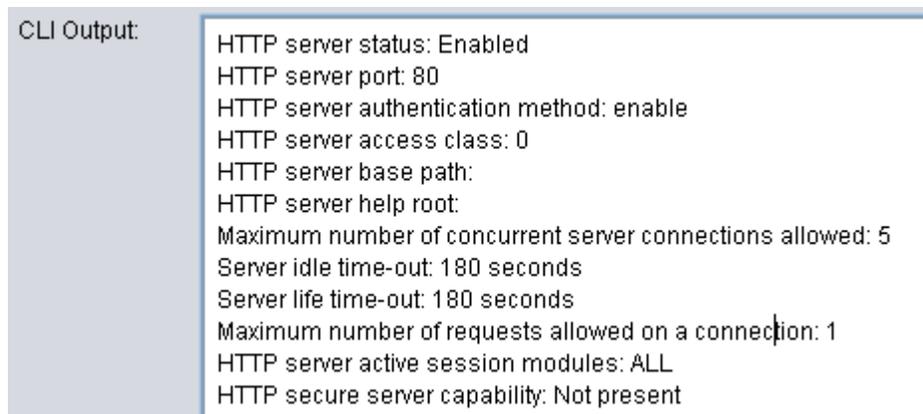


Figure 1.1.3

2.2 Develop Data Model

Highlight each segment of text to be extracted into XML in the final output. This is achieved by first highlighting the section of text in the CLI Output text area that will be used as the property name. Then, right-click on the highlighted text, move down to the Add Property option, and select the Name option (see figure 1.1.4).

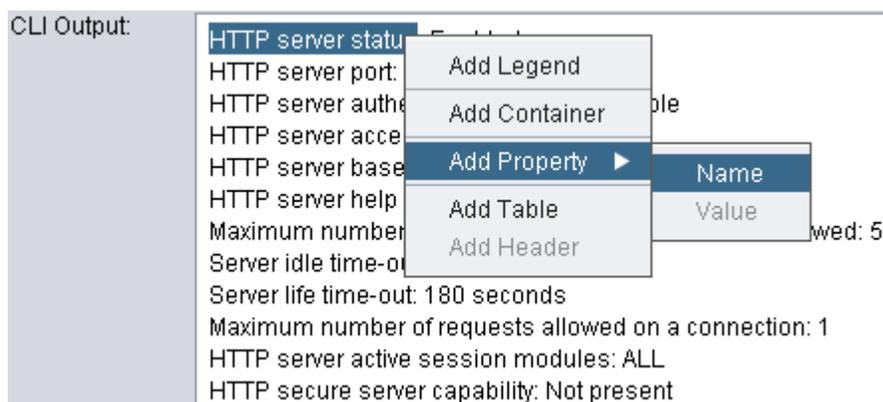
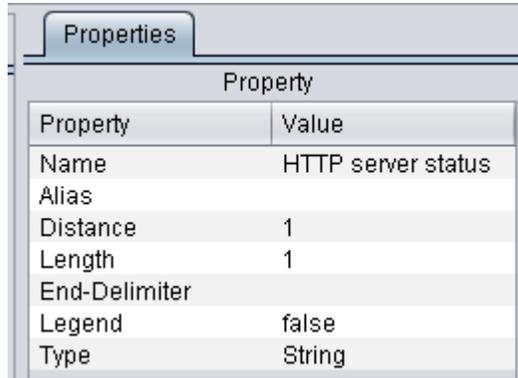


Figure 1.1.4

After choosing the Name option from Add property, information on the new Property appears on the right hand side under the Properties tab (see figure 1.1.5)



Property	Value
Name	HTTP server status
Alias	
Distance	1
Length	1
End-Delimiter	
Legend	false
Type	String

Figure 1.1.5

Next, specify what value this new Property will capture when processed by a Cisco IOS device by highlighting the text in the CLI Output text area, right-clicking on the highlighted value, move down to Add Property, and select the Value option (see figure 1.1.6). Now Rule Editor knows what value to retrieve for the newly defined property.

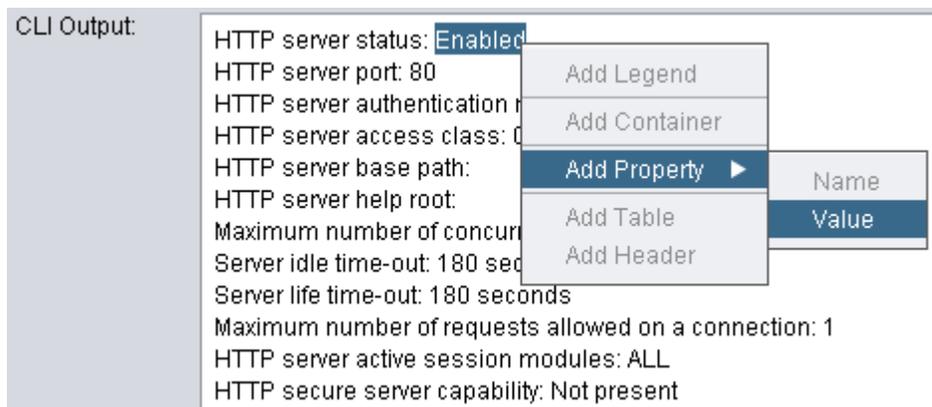


Figure 1.1.6

Continue this process of highlighting and setting properties and their values until all relevant text segments in the CLI Output text area have been covered.

2.3 Generate Specification File

Once all properties and their values have been defined, Rule Editor is ready to generate a Rule specification file. To generate a specification file, click on the Generate XML Spec button located in the top left corner of the application under the File menu. After clicking the Generate XML Spec button, Rule Editor will display a new spec file of processing rules based on the properties defined in the data model (see figure 1.1.7).

```
<?xml version='1.0' encoding='utf-8'?>
<ODMSpec>
<Command>
<Name>show ip http server status</Name>
</Command>
<OS>ios</OS>
<DataModel>
<Container name="ShowIpHttpServerStatus">
<Property name="HTTP server status" distance = "1" length = "1" type = "String"/>
<Property name="HTTP server port" distance = "1" length = "1" type = "String"/>
</Container>
</DataModel>
</ODMSpec>
```

Figure 1.1.7

To check what XML output the specification file produces with the original command output, click on the Test Spec Local button. Rule Editor takes the current specification file, applies the rules from the file to the command output in the CLI Output text area, and displays the resulting XML output (see figure 1.1.8).

```
<?xml version='1.0' encoding='utf-8'?>
<ShowIpHttpServerStatus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<HTTPServerStatus>Enabled</HTTPServerStatus>
<HTTPServerPort>80</HTTPServerPort>
<Info>HTTP server authentication method: enable</Info>
<Info>HTTP server access class: 0</Info>
<Info>HTTP server base path: </Info>
<Info>HTTP server help root: </Info>
<Info>Maximum number of concurrent server connections allowed: 5</Info>
<Info>Server idle time-out: 180 seconds</Info>
<Info>Server life time-out: 180 seconds</Info>
<Info>Maximum number of requests allowed on a connection: 1</Info>
<Info>HTTP server active session modules: ALL</Info>
<Info>HTTP secure server capability: Not present</Info>
</ShowIpHttpServerStatus>
```

Figure 1.1.8

In Figure 1.1.8, HTTPServerStatus and HTTPServerPort are XML tags derived from the rules specification file generated in Rule Editor. The Info tag is Rule Editor's way of displaying data that did not match any rule in the specification file.

3. Defining Table Rules

3.1 Table Command Output

Many of the IOS commands on a Cisco IOS device return output in the form of a table. In such cases, it's not enough to identify properties and their values. When processing table output, the Rule specification file must specify each column in the table along with the start and end position of each column (see figure 3.1.1 for an example of table output).

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	10.1.43.1	1	0012.01c6.02c3	ARPA	GigabitEthernet0/1
Internet	10.1.43.55	-	0019.306b.221b	ARPA	GigabitEthernet0/1
Internet	10.1.43.210	2	0022.5583.9eb2	ARPA	GigabitEthernet0/1
Internet	10.1.43.254	32	0012.4349.4721	ARPA	GigabitEthernet0/1

Figure 3.1.1

The table output above comes from the show arp command. The column names that would be used in the Rule Editor data model are Protocol, Address, Age (min), Hardware Addr, Type, and Interface

3.2 Building A Table Model

First, create a new project (refer to section 2 Rule Editor Work Flow) and copy the table output from the command into the CLI Output text area. After copying the table output into the CLI Output text area, right click on the Model file under the Projects Explorer tab, select Add Table, and give the table a name (see figure 3.2.1).

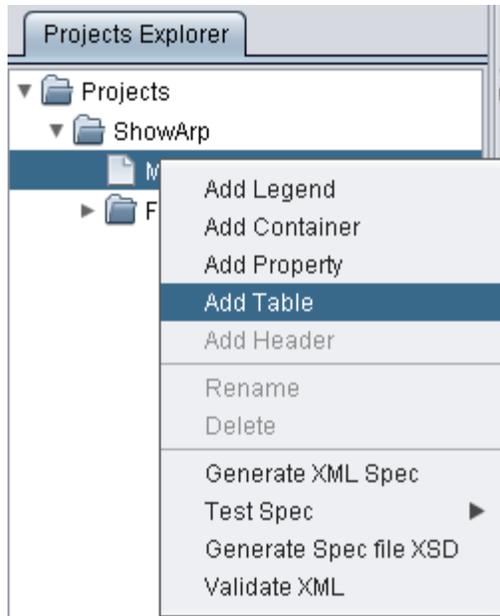


Figure 3.2.1

When the table shows up under the Projects Explorer, highlight the first column name in the command output, right click on the highlighted column name, and select the Add Header option (see figure 3.2.2).

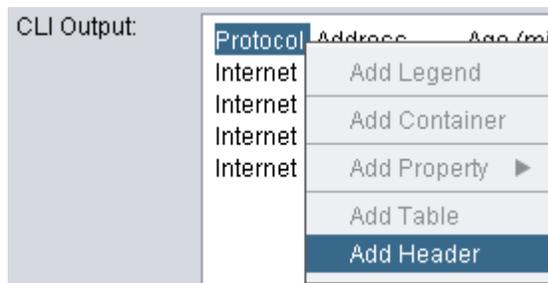


Figure 3.2.2

After selecting the Add Header menu item, a Table Header property will appear under the Properties tab on the right hand side (see figure 3.2.3). The two most important values in any Table Header property are the start and end values. These values tell a Cisco IOS device the width of a column in terms of character length. If start is 0 and end is 8, the width of the column is 8 characters long.

Table Header	
Property	Value
Name	Protocol
Alias	
Start	0
end	8
Nullable	false
Wrappable	false
Legend	false
Type	String

Figure 3.2.3

Continue this process until all table headers are defined in the current model. When the model is ready, generate the specification file and test the file against the command table output.