Automated Evaluation of a Network Device Configuration

Z. Csengődy, D. Macko and K. Jelemenská
Faculty of Informatics and Information Technologies
Slovak University of Technology
Bratislava, Slovakia
dominik.macko@stuba.sk

***Abstract*—Process automation increases quality of human everyday life and boosts work effectiveness. Such benefits are also achievable while testing and evaluating configurations of manageable network devices (e.g. routers or switches). This is especially crucial in education, where many students configure many devices and their teacher must check whether the configurations are correct. This paper is focused on such a problem. We propose a new method, along with a corresponding tool, that is able to load the configurations to be tested and compare them against correct configurations, provided by a teacher. It does not only compare the text in configuration files, but also takes into account structural characteristics of configuration commands. The proposed tool also evaluates the configurations in terms of percentage and obtained points, and thus it can be also used for student exams.**

I. Introduction

It is undeniable that automation makes the processes more efficient in terms of time and costs. It also reduces the possibility of introduction of a human error. The process automation is also very important in education sphere. There is a need to automate as many education processes as possible, for a teacher to focus on more important aspects instead of routine activities, which enables the public money to be efficiently used. It is also important to objectively evaluate students works, thus the same results be valued evenly, which reduces discrimination.

In this area, there are many simulation tools or testing frameworks used to alleviate such problems. However, students must also increase their working skills with real equipment and there is little automation involved to evaluate such results – i.e. whether they used the equipment correctly and obtained the expected results. We must find a way to automate also such evaluation processes.

This is also the case when teaching students to configure manageable active network devices, such as routers or switches. Due to time optimization and difficulty for a teacher to evaluate multi-device configurations of many students in a study group, the students often work in a simulation environment, such as Cisco Packet Tracer simulation tool [1]. Using such a tool the configurations can be easily and quickly checked; however, students lose touch with the problems occurring in real devices only (e.g. console connection, cabling problems, wrong device, or hardware faults).

In this paper, we deal with a problem when students configure real hardware network devices and the configuration must be checked for correctness. Usually, this is done manually by their teacher, which knows what the correct configuration should be. However, it takes time and makes unnecessary waiting periods during the class, since the teacher evaluates a single student at a time (i.e. students must wait for their turn). This process can be optimized using the proposed method for automated evaluation of network-devices configurations. It enables the teacher to beforehand prepare a golden-model configuration, against which the students' configurations will be checked automatically. Thus, the time during class can be spent more efficiently.

The paper is organized as follows. In the next section (Section 2), the related works are discussed, concerning automated evaluation of network-devices configurations. Section 3 is the main part of the paper, which describes the proposed method and the tool for automation of the configuration-evaluation process. In Section 4, we provide some results from experimental testing of the method and the tool, using the actual students-configured scenarios. And finally, we conclude the paper and discuss further work in this area (Section 5).

II. Related Works

This section is divided into two parts: the analysis of existing works that help to parse configuration files, and the analysis of existing works for comparison and evaluation of configuration files.

A. Configuration Files Parsing

Configurations of Cisco network devices operating under the Cisco IOS (Internetwork Operating System) are stored in text-based configuration files. Thus, evaluation of configuration files requires text parsing and analysis of the parsed information. A configuration file contains hierarchically ordered configuration commands with a specific structure.

As a help to work with configuration files, one can use, for example, the conftodict [2] Python module. It is able to convert content of a configuration file to a Python dictionary, which is based on a key-value concept. The key is a parent command and the value is a list of the corresponding children commands (i.e. one hierarchical level lower than the parent, but in its context). It is then easier to find specific commands in specific sections of the configuration file; therefore, it can be also used for comparison of configuration files and their evaluation.

This is an accepted version of the published paper:
Z. Csengődy, D. Macko and K. Jelemenská, "Automated evaluation of network device configuration," 2018 16th International Conference on Emerging eLearning Technologies and Applications (ICETA), Stary Smokovec, 2018, pp. 99-104.
doi: 10.1109/ICETA.2018.8572175
URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8572175

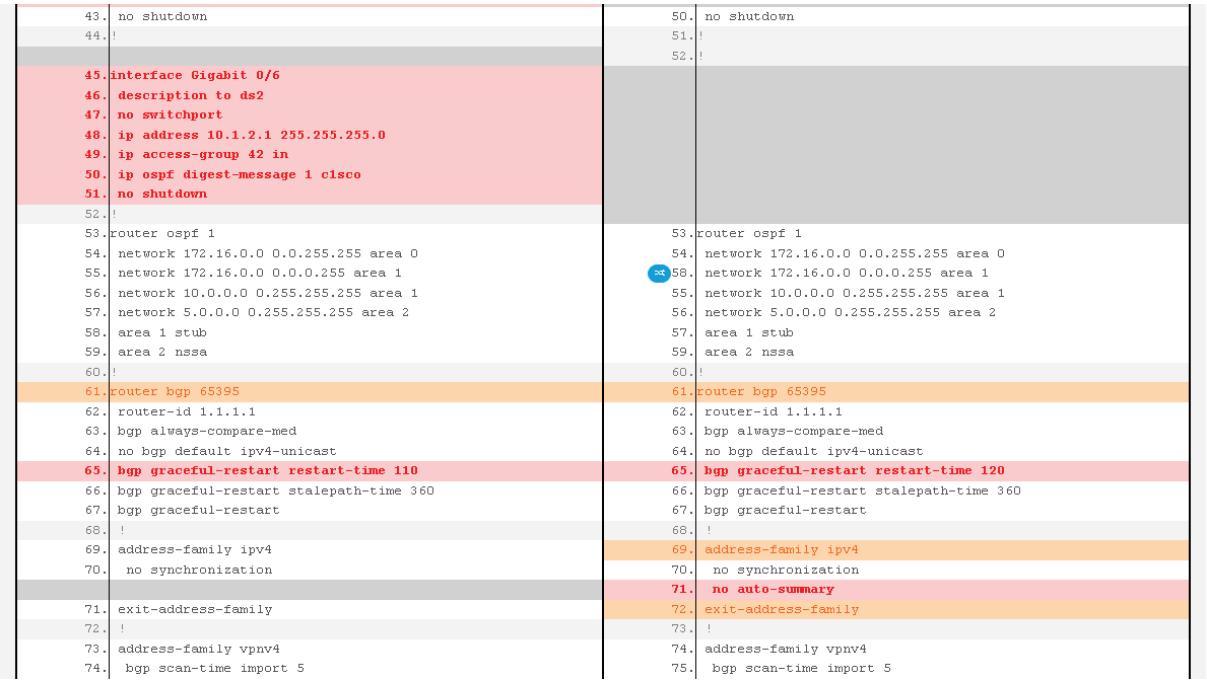

Figure 1. An output example of the Cisco Configuration Diff tool [4].

Another approach is proposed in [3], which is able to adapt to changes in the configuration language (i.e. updates), thus reducing overhead of manual modification of the parsers. The proposed parsing consists of two phases. In the first phase, command types are used to assigned parsed commands into syntactic structural categories and then grouped into a tree-like data structure. In the second phase, the analyzed configuration is transformed into the XML (eXtensible Markup Language) form. Unrecognized commands by the second phase are processed by a script, which accordingly updates the first phase of parsing, and thus is able to adapt to language changes.

B. Configuration Files Comparison and Evaluation

A tool called Cisco Configuration Diff [4] solves the false-positives problem with the usual text-comparison tools, when used on configuration files. Since the tool understands the hierarchical structure of configuration files, it is able to reorganize and align the corresponding sections of the two compared configuration files. An example of possible output from the tool is illustrated in Fig. 1. The missing or different lines are marked by the red color, the beginning and the end of a section with a red line is marked by orange color, and the blue icons mark the reordered lines. The tool is useful for a fast comparison of two configuration files; however, it is impractical for comparison of a higher amount of students' configuration files.

A tool Contextual Configuration Diff Utility [5] offers very similar capabilities of identifying differences between two compared configuration files and identifying lines with different order. However, this tool operates directly within Cisco integrated file system, and is thus useful for comparison of different versions of configurations in a single device. An example of the tool output is illustrated in Fig. 2. Marks at the beginning of a line represents whether the command was added in the second configuration file ("+"), whether the command

```
+ip subnet-zero
+ip name-server 10.4.4.4
+voice dnis-map 1
 +dnis 111
interface GigabitEthernet1/0/0
 +no ip address
 +shutdown
+ip default-gateway 10.5.5.5
+ip classless
+access-list 110 deny ip any host 10.1.1.1
+access-list 110 deny ip any host 10.1.1.2
+access-list 110 deny ip any host 10.1.1.3
+snmp-server community private RW
-no ip subnet-zero
interface GigabitEthernet1/0/0
 -ip address 10.7.7.7 255.0.0.0
-no ip classless
-snmp-server community public RO
```

Figure 2. An output example of the Contextual Configuration Diff Utility [5].

available in the first configuration file is not present in the second configuration file ("-"), or whether some lines in the second configuration file have different order than in the first file ("!"). Lines without any mark specify context of subsequent commands (within the hierarchical structure of the configuration file).

In comparison to the previous two tools, the Cisco Packet Tracer simulator [1] is able not only to compare the configurations of multiple devices against predefined ones, but also to automatically evaluate them – i.e. to assign points for correct commands (i.e. commands are the same as the predefined commands) and to compute progress in terms of percentage. Thus, it is usable also for testing purposes. This is achievable by creating a Packet Tracer activity using the Activity Wizard (illustrated on the left in Fig. 3). In the Answer Network section of the Activity Wizard, a teacher can select which commands will be valued and assign point to each command (or function). After a student configures devices, the Check Results button opens a window shown on the right side in

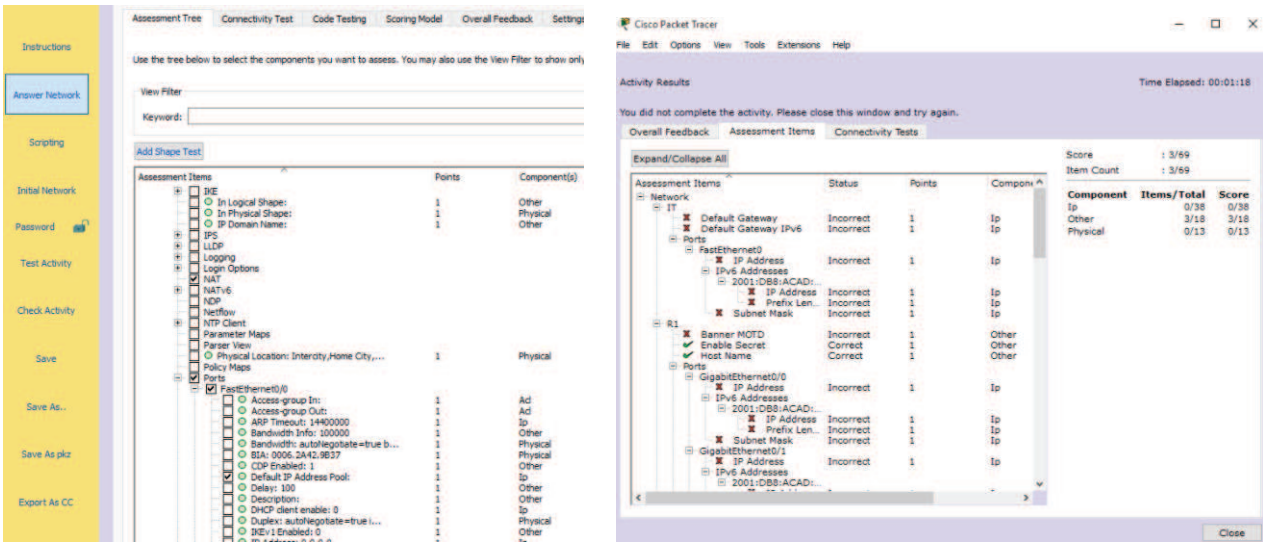

Figure 3. Cisco Packet Tracer Activity Wizard.

the figure. If allowed by the teacher, the student can scroll the valued commands for each device and see if they were configured correctly. Since Packet Tracer is a simulator, it supports only a subset of Cisco IOS commands. Also, it does not enable to evaluate a real hardware device configuration; therefore, another configuration-evaluation approach is required.

III. Automated Configuration Evaluation Method

Our goal is to automate the process of checking and evaluation of students' configurations of real hardware network devices. We have analyzed multiple existing tools and approaches; however, none of them fulfills all of our needs. Therefore, a new method is required, for which we have stated the following requirements:

- Loading of the golden-model configuration files for multiple devices.
- Loading of the configuration files to be checked and evaluated.
- Visualization of the golden-model configuration commands, preserving the hierarchical structure.
- Enabling to specify multiple alternative configurations of the same function in the golden model.
- Assigning the points to the commands of a selected configuration file from the golden-model files.
- Saving the settings for each golden-model configuration file.
- Comparison and evaluation of the configuration files to be checked against the golden model and according to the settings.
- Creating the evaluation report stating which commands were erroneous or missing in the checked configuration and stating the overall evaluation result in terms of points, percentage, and grade.

In addition to these functional requirements, we focused on development of a clear, simple, and intuitive graphical user interface of the tool, without redundant information. Thus, a teacher should be able to use the tool without a need to read the user guide.

A. *The proposed method*

After the golden-model configuration files are inserted, they are analyzed and checked, whether they contain additional configuration files containing an alternative configuration of some function. These files are identified using their name, which must include a specific suffix "_add". The prefix of the additional configuration files must correspond to the name of some of the main configuration files (usually the hostname of the device, to which it belongs). The alternative configuration is then appended into the corresponding main configuration, and it is afterwards treated as a single-file configuration. Such functionality is missing in all existing methods, making our method unique. Many functions of a network device can be configured using multiple ways (e.g. filtering using the access control lists), but the results is the same. Thus, students can choose the way they fulfill the task, enabling a teacher to prepare goal-oriented exams (not specifying exactly what commands the students have to use).

We also propose to use a variable configuration in the golden model, which simplifies the specification of alternative configurations even more. This functionality is based on usage of a question mark ("?") in a golden-model configuration file, which can be interpreted as any string if used as a word (i.e. separated by a white-space character). If "?" is used inside a string, it represents a single variable character. Several examples of usage of the proposed variable configuration are provided in Table 1.

TABLE I.
Example of Variable Configurations Usage

Golden-model configuration	Checked configuration
router ospf ?	router ospf 10
line vty ? ?	line vty 0 4
network ???.???.?.0 0.0.0.0 area 0	network 192.168.1.0 0.0.0.0 area 0
mpls ldp router-id Loopback? ?	mpls ldp router-id Loopback0 force
mpls ldp router-id Loopback? ?	mpls ldp router-id Loopback7

The first column represents the variable configuration in the golden model and the second column represents a configuration that matches the golden model. Multiple configuration commands can match a single command when variable configuration is used, as shown in the last two rows.

The configuration evaluation cannot be done without settings of commands points. After the loading of the golden-model configuration files, the point-values can be assigned to each command for a selected configuration file (including alternative configuration commands). There is also possibility to assign a point-value for a hierarchical section, which means that the points will be counted only if all subcommands (i.e. all children commands of the corresponding parent command) will be correct. This is also a unique solution, which gives a teacher a flexibility and opportunity to evaluate configurations with different granularity.

Before the comparison of configuration files, there is a verification step ensuring that the number of configuration files meets the number of golden-model main files. Also, it is verified that each configuration file to be checked has a pair file in the golden-model configuration (using file names). After the successful verification, all the configurations are transformed into the XML form, which enables to easily preserve the hierarchical structure. The golden-model configurations are then sequentially processed, and points are counted if the commands are found in the checked configurations, respecting the context (the commands are looked for in the corresponding section). If a golden-model command is not found (or is incorrect) in the checked configurations, this situation is reported after the processing is finished.

B. The implemented tool

We have implemented the proposed method into a simple and easy-to-use tool. The implemented graphical user interface is illustrated in Fig. 4. As shown, it consists of two windows. The initial primary window is provided on the left side. The configuration files are inserted into the tool using the drag-and-drop method. The golden-model configuration files must be dropped in the *Control File(s)* section, the configuration files to be checked must be dropped in the *Test file(s)* section of the window. After the files are inserted into the tool, the points for commands of the golden-model configurations have to be assigned.

By clicking the icon in the top-left corner, a user can open the settings window (shown on the right in the figure). Using the combo-box, the user is able to switch between individual configuration files of the golden model. After the changes are made in the point-values of the commands of a configuration file, the settings must be saved (individually for each file). Again, using the icon in the top-left corner, the user switches to the primary window of the tool.

Using the *Compare* button, the user starts processing of the configuration files. After the configuration files to be checked are evaluated, the results are displayed in the bottom section of the primary window.

After starting the tool, a log file is created, which contains all messages and possible exceptions, using the following syntax:

Syntax: # Priority_Class_Method_(Index) – Exception – Line: Number

The file reporting the results is named "results.txt" and it consists of sections, marked by "## *Hostname_of_nework_device* ##". In each section, there are erroneous/missing configuration commands for the corresponding device reported. The lines reporting alternative configurations are marked by "-- *Additional Configuration*". At the end of the file, there is the overall result reported in the form of:

Obtained_points/All_points – percentage

An example of the results file contents is provided in Fig. 5.

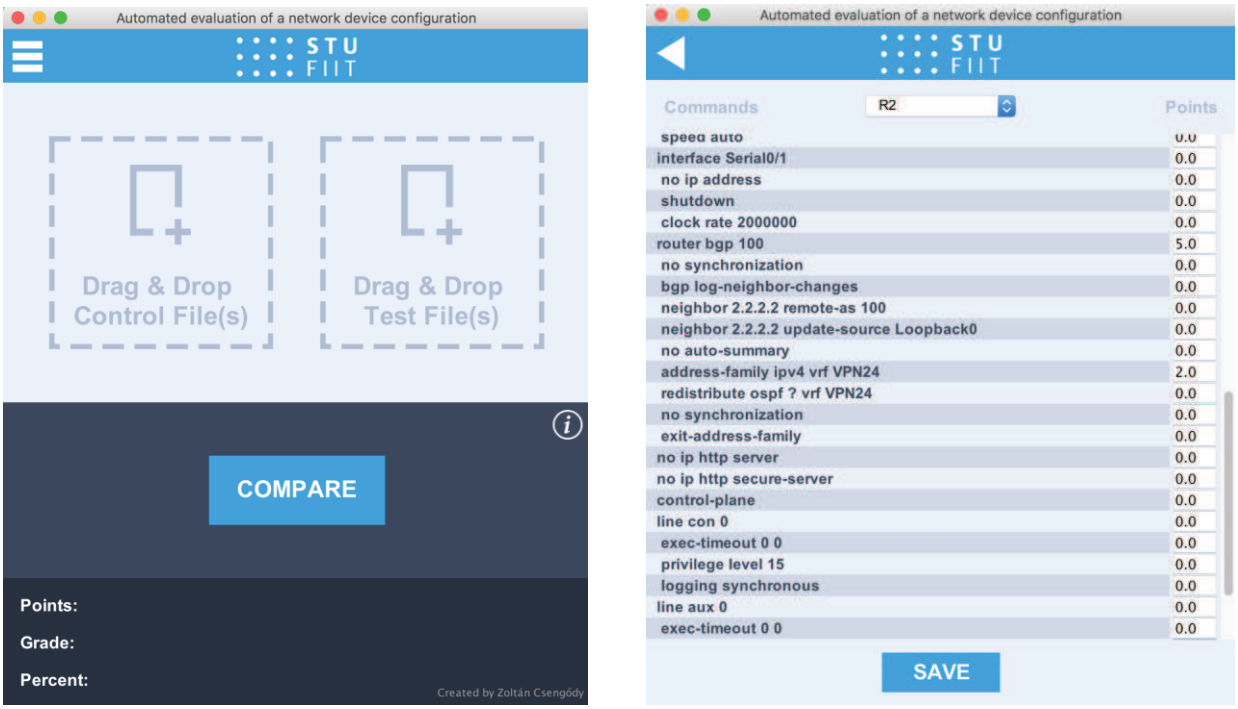

Figure 4. Graphical user interface of the proposed tool.

```
## P1 ##

## P2 ##

## PE1 ##

redistribute ospf ? vrf VPN24 metric ? --
Additional Configuration

## PE2 ##

no auto-summary

neighbor 1.1.1.1 route-map Mapa out
redistribute eigrp ?

ip prefix-list List seq 5 permit
192.168.0.0/16 le 29
route-map Mapa permit 10
match ip address prefix-list List

no auto-summary -- Additional Configuration

## 11.5/16.0 - 71% ##
```

Figure 5. An example of the generated file "results.txt".

IV. Experimental Results

During and after development of the tool, we have done application tests to verify functionality of the tool. After verification of the functional aspects of the tool (and thus the method too), we have used the tool to evaluate students' configurations, previously graded by a teacher. Then we have compared the obtained resulted points provided by the manual evaluation by a teacher and the automated evaluation using the proposed method. In these experiments, we have not used the proposed variable configurations, since it was neither used by the teacher during manual evaluation. Thus, the students' configuration should match exactly the golden-model specification.

The results are provided in Table 2. There were two test scenarios used (*Test1* and *Test2*), and for each, configurations of five students were evaluated ($A - E$). The maximum points achievable in the test were 20; however, there were two points for speed and two points for filters (which could not be evaluated automatically, since variable configurations could not be used). Thus, the maximum for automated evaluation was 16 points. When comparing the results, we can see that the difference between automated and manual evaluation is within these four points.

The manual evaluation has not counted some mistakes as errors, such as when at one place was the configuration command correct, but the same command was configured incorrectly at some other place. However, such a configuration was not functionally right; therefore, the points should not be assigned to the student. Thus, the proposed automated evaluation is accurate, more objective, and reduces discrimination – all students are evaluated in the same manner.

The manual evaluation took about 5–10 minutes for each student. The results from automated evaluation were provided very fast, like in few seconds. Thus, the speed-up is significant and very useful for a teacher.

V. Conclusions and Further Work

In this paper, we have been dealing with a problem of evaluation automation for configurations of real hardware network devices. We have proposed a method and implemented a tool that is able to parse configuration to be checked and compare them against a golden-model configuration (a correct configuration). Comparison results are used to evaluate the checked configuration and assign points and grade; thus, it is especially useful for evaluation of students' configurations in education sphere.

By experimental verification, we have shown that the proposed automated evaluation provides accurate results and reduces the discrimination – the tool treats the students evenly and the evaluation is objective. The speed-up achieved by automation of the evaluation process is significant. The results from the tool are obtained immediately; thus, a teacher can grade the students during the class without time problems.

Compared to the existing methods and tools, we have proposed unique possibility to specify multiple correct configurations of the same function. Also, we have increased flexibility and simplified the specification of the golden-model configuration by the proposed variable configurations.

In the future, we will focus on limitations of the proposed tool. It could automatically load configurations of multiple students and evaluate them, instead of inserting them manually for each student. It would also

TABLE II.
Comparison of Manual and Automated Student Tests Evaluation

Student	**Test 1 [points]**		**Test 2 [points]**	
	Manual evaluation	***Automated evaluation***	***Manual evaluation***	***Automated evaluation***
A	15	11.5	13	9.5
B	20	16	17.5	16
C	11.5	7.5	11.5	11
D	7.5	4	15	8
E	12.75	11.5	7.5	8
Average	13.35	10.1	12.9	10.5
Average difference	3.25		2.4	

require modification of grading scheme. Further work in this area can be also oriented towards securing the golden model by encryption, authentication, and authorization. It would enable students to check their configurations against the provided golden model themselves; however, they could not see the golden-model configurations contents or modify points-assignments.

ACKNOWLEDGMENT

This work was mainly supported by the Slovak Cultural and Educational Grant Agency (KEGA 011STU-4/2017 – "Update of computer networks curricula based on needs of practise") of the Ministry of Education, Science, Research and Sport of the Slovak Republic. It was also partially supported by the Slovak Scientific Grant Agency (VEGA 1/0836/16) and the Slovak Research and Development Agency (APVV-15-0789).

REFERENCES

[1] K. Kniewald, F. Jakab, and J. Janitor, "Visual learning tools for teaching/learning computer networks: Cisco Networking Academy and Packet Tracer," in *2010 Sixth International Conference on Networking and Services (ICNS)*, 2010, pp. 351-355. DOI: 10.1109/ICNS.2010.55.

[2] B. Searle, "conftodict: Python module to convert Cisco IOS config to a python dictionary," GitHub, 2016. [Online]. Available: https://github.com/bobthebutcher/conftodict.

[3] D. Caldwell, S. Lee, and Y. Mandelbaum, "Adaptive parsing of router configuration languages," in *2008 IEEE Internet Network Management Workshop (INM)*, 2008, pp. 1-6. doi: 10.1109/INETMW.2008.4660333

[4] C. Dessez, "Cisco Configuration Diff," 2016. [Online]. Available: https://community.cisco.com/t5/network-architecture-documents/cisco-configuration-diff/ta-p/3157684

[5] Cisco Systems, *Managing Configuration Files Configuration Guide, Cisco IOS Release 15M&T*. San Jose, CA: Cisco Systems, Inc., 2015.