OSS Interface for HP Service Activator
Paulo Jorge Pontes Martins

Abstract—Communications Service Providers (CSPs) are taking too much time to deploy new services, mostly due to its hard network infrastructure, which leads to an increasing demand for more dynamic and scalable networks. Hewlett-Packard (HP) and its product HP Service Activator (HPSA) are a reference on the automation of Operations Support Systems (OSS) fulfillment for CSPs, working as an abstraction layer between the network and the rest of the OSS. Network Configuration Protocol (NETCONF) is the most recent network management protocol standardized by Internet Engineering Task Force (IETF) and one of the main growing standards in the network management market. This project presents an analysis to the most common used network management protocols used by CSPs for services provisioning and activation and implements a solution which enables CSPs to use HPSA with a NETCONF southbound interface, supporting all the operation defined in the standard and the possibility of concurrently manage more than one device. The final tests confirm not only the usability of this solution to interact with NETCONF enabled network devices but also the advantages of using this plug-in instead of other available solutions.

Index Terms—NETCONF; HP Service Activator; Communications Service Providers; Network Management; Activation; Provisioning

1 INTRODUCTION

The Communications Industry is facing a major challenge where the only constant seems to be the “change”. Communications Service Provider (CSP) all over the world are facing great challenges to remain profitable, to meet the high customer expectations and to confront the fierce competition.

Well defined business processes and the right automation tools for these processes have been an important factor on who succeeds in this kind of market. For this reason, the TeleManagement Forum (TMF) created a business framework [1], identifying the required and/ or recommended business and operations processes to successfully run a “services-based” business.

In the Operations Support Systems (OSS) “Services Fulfillment” area of TMF’s business framework, the need to manage network and service platforms or equipments has generated significant discussion in entities like Internet Engineering Task Force (IETF) and Distributed Management Task Force (DMTF) which are responsible for creating standards for the related technologies. The evolution of the services network characteristics, such as its dimension, the number of devices, the wide range of different suppliers and even the growing diversity of services, have changed the management requirements and consequently created a demand for new technologies to manage those networks [2].

Over the years, several problems related with the lack of security and scalability to manage large infrastructures and the huge volume of information were pointed as weaknesses to the most commonly used Network Management protocols, such as Command Line Interface (CLI) and Simple Network Management Protocol (SNMP) [3]. In order to overcome the current problems the IETF is developing a new technology named Network Configuration Protocol (NETCONF) which uses Extensible Markup Language (XML) and Remote Procedure Call (RPC) to wrap its protocol messages, requiring transport protocols able to ensure authentication, integrity and confidentiality.

Hewlett-Packard (HP), a lead supplier in automated solutions for OSS Fulfillment [4],
deployed HP Service Activator (HPSA), the solution responsible for services Provisioning and Activation. HPSA provides CLI and Hypertext Transfer Protocol (HTTP), two generic plugins to communicate with network elements in order to edit their configuration. These plugins are however “proprietary” in terms of the “methods/commands” used for the Provisioning and Activation processes.

Everyday, new Cloud, Mobile, Internet Protocol television (IPTV) or even Over-the-top content (OTT) services appear in the Information technology (IT) world, however, the CSPs processes used to configure network infrastructures are not as fast or error free as they should in order to keep up with this evolution. Hence the importance of using the right tools or protocols.

This work comes from the need and the demand from CSPs to use NETCONF as a network configuration protocol for their Activation and Provisioning services. That demand reached HPSA’s development team, who felt the need to support that protocol in their Provisioning and Activation tools. The contribution of this work is two-fold. First, it contains an analysis of the current state of art of services Provisioning and Activation processes in CSPs, including their positioning in the standard structure of a CSP and the main tools and network management protocols used for that purpose. Second, it presents a tested solution for CSPs, enabling them to use NETCONF as a network management protocol for services Activation which, taking into account the age of the NETCONF standard – RFC 4741 [5] – and the currently available tools, is something not very common, but something with a growing hype in research and in industry.

The objective of this work is to integrate NETCONF and HPSA using the tools provided by HPSA for its customization. This document describes the importance of HPSA in the context of the current CSPs business model, the most relevant management protocols presently being used and why NETCONF is a suitable solution for HPSA. Therefore, and with that purpose, a new HPSA Plugin was designed, implemented and tested, following standard integration, conformance testing suites and methodologies from HP. The design and development of the prototype is described in detail in this document, as well as a description of the implemented solution and its evaluation.

2 BACKGROUND

2.1 Enhanced Telecom Operations Map

In 2003 the TeleManagement Forum (TMF) released the Enhanced Telecom Opererations Map (eTOM) Business Process Framework as an International Telecommunication Union (ITU) standard [1], and even though it is not mandatory but a recommendation, it has been adopted by most CSPs [6]. The eTOM identifies the required business processes to run a service-focused business, and divides those processes recursively in sub-processes, until it is clearly distinguishable the “product management” aspects from the “network operations” aspects. The initial Level-0 process groupings view of the framework defines a structuring of the overall enterprise into the three major areas of Strategy, Infrastructure and Product, Operations and Enterprise Management.

2.2 Network Management

Despite the vast range of network management standards and protocols, most of the network management systems are still very dependent on the human interaction. A large number of Router vendors have their own configuration languages and most of them are configured through CLI or Web-based technologies. It is not an issue using manual configuration, if a CSP intends to automate the configuration processes, but it might be a problem if no automation is considered, since manual configurations can lead to errors, jeopardizing the proper functioning of the entire network.

For a long time, several operators had the opinion that the developments made in IETF were not taking into account the operator’s needs, especially for configuration management. Given this generalized opinion, several meetings were organized to discuss the Requirements for Configuration Management of IP-based Networks [7].
On June 4 2002, some members of the Internet Architecture Board (IAB) held a workshop where operators and protocol developers had the chance to exchange their opinions about the IETF focus on future work regarding network management. That workshop had two main goals: Name a list of relevant network management technologies and identify their strengths and weaknesses; Understand the operator’s needs based on the already existing solutions for network management. The outputs of that workshop can be found on RFC 3535 [8] where the analysis of the most relevant network management technologies originated a set of requirements which were later used as a base for the development of NETCONF.

2.3 NETCONF

NETCONF tries to solve the limitations found with the other above mentioned protocols, by addressing the operator’s necessities. Some of the key requirements that a new network management solution should fulfill in order to meet the operator’s needs are the following [9]:

1) There must be a clear distinction between configuration data and operational data. Operational data is the additional data on a system that is not configuration data such as read-only status information and collected statistics.
2) It must provide primitives to prevent errors due to concurrent configuration changes. Either a lock or a conflict resolution mechanism are necessary.
3) There should exist primitives to apply configuration changes to a set of network elements in a robust and transaction-oriented way.
4) There must be a way to distinguish between several configurations, and devices should be able to hold multiple configurations.
5) There should be a distinction between candidate configuration and running configuration.
6) It should be clear about the persistence of configuration changes.
7) Must report configuration change events.
8) Full configuration dump and a full configuration restore must be supported appropriately.
9) A configuration management protocol must represent configuration state and operational state in a form enabling the use of existing tools for comparison, conversion, and versioning.

NETCONF was first published as a Standard on December 2006 in RFC 4741 [5], by the NETCONF Working Group within the IETF, being later overwritten in RFC 6241 [10]. NETCONF presents a four layer architecture, where each layer plays a very specific function:

1) The **Transport** layer provides communication between the client (manager) and the server (managed object).
2) The **Messages** layer defines the reply-request mechanism to encode RPCs and Notification messages.
3) The **Operations** Layer defines the set of operations encoded in XML and invoked as RPC methods.
4) The **Content** layer declares the data exchange in NETCONF operations.

NETCONF is connection-oriented, meaning that it requires a persistent connection between the pairs. The connection must grant authentication, integrity and confidentiality, and so NETCONF delegates to the transport layer the responsibility of granting the appropriate levels of security and confidentiality. Even though NETCONF is defined as being session-layer and transport independent, it is mandatory to any implementation the support of the Secure Shell (SSH) connection using the IANA-assigned Transmission Control Protocol (TCP) port 830 and the subsystem netconf [11].

The NETCONF communication model is based on RPC. Both peers use the elements `<rpc>` to frame the requests sent by the client to the server and `<rpc-reply>` to frame the server’s answer. The `<rpc-reply>` may contain the `<ok>` element, informing that no error occurred during the execution of that request or an `<rpc-error>` with a set of information about the detailed errors detected.

Every NETCONF implementation must define its Capabilities. A Capability is identi-
fied by an unique Uniform Resource Identifier (URI) and allows the clients to discover the set of operations supported by the servers allowing the client to adjust itself to exploit all the resources exposed by the devices. During the session establishment, each peer (client and server) sends a <hello> element with the list of capabilities. There are more capabilities, but for the purpose of HPSA, and in the scope of this project the focus will be on the capabilities identified in the main RFCs of NETCONF, more specifically, the ones related with configuration data.

Capabilities are advertised in messages sent by each peer during session establishment. When the NETCONF session is opened, each peer (both client and server) must send an <hello> element containing a list of that peer’s capabilities. Each peer must send at least the :base:1.0 capability.

2.3.1 YANG

NETCONF allows the access to native capabilities of devices in a network, defining methods to manipulate the configuration of those devices and to retrieve operational data about that device. YANG is a data modeling language used to model NETCONF’s content and operations layers and it is defined in RFC 6020 [12].

A YANG module works like a contract between the client and the server. After both peers agree which modules they will use, the client is aware of how to create valid information to be transmitted to the server and it understands the server’s replies. Using NETCONF and YANG together, standard modules may be defined, allowing the interoperability between devices from different manufacturers and allowing also those devices to expose their unique capabilities.

2.3.2 Current Implementations

NETCONF is a recent protocol, yet it is being largely accepted by network administrators. It has the support of some of the world’s greatest network device vendors like Cisco, Juniper, Ericsson and Oracle, and there are already some open-source and proprietary implementations on the market such as netopeer. An open-source implementation developed by CESNET, based in the a NETCONF library written in C called libnetconf or Java NETCONF Client (JNC) [13], a NETCONF server and client implementation in JAVA or even android NETCONF client solution named netconf4android.

2.3.3 NETCONF Evolution

The demand for agility in services delivery of CSPs is driving the interest in Software Defined Networks (SDN) and Network Functions Virtualization (NFV) technologies [14], making them one the most discussed topics in the 2015 edition of the Mobile World Congress, which takes place every year in Barcelona, Spain. NFV is an network architecture which proposes the virtualization of network functions that currently runs on commodity servers, something that was really difficult until now given the low processing power of existent CPUs and Network interfaces. SDNs is a network architecture which separates the infrastructure layer from the controller layer which works as a mediator for applications who intend to interact with network devices and translates those application requests in network configuration. Controllers tell network devices how to behave.

NETCONF is one the main protocols proposed for the interaction either from SDN controllers to network devices, as from the application layer to the controller layer. Some well known SDN controllers, like OpenDaylight and Ryu, support NETCONF as an interface. For both technologies SDN and NFV, NETCONF is one of the main configuration protocols being discussed and implemented, given its standardization for all different kind of vendors and implementations.

2.4 HP Service Activator

HP Service Activator (HPSA) 6.1 is a customizable product from HP released in 2008 that performs tasks to activate services offered by providers of converged technologies
MARTINS

and CSPs. HP Service Activator (HPSA) is usually deployed by CSPs to automate and secure repetitive jobs, guaranteeing error-free processes and focusing on the processes designated in eTOM as Service Configuration and Activation and Resource Provisioning [15].

The revenue from the Activation was more than 450M US Dollars in 2009 and HPSA is one of the most used Provisioning and Activation solutions used in the market [4]. HPSA is recognized for its scalability and its user interface which eases the usability. The generic Workflow engine of HPSA can be used to model complex activation processes and its robustness makes it a high-reliable solution. The main goal is to reduce the time-to-market of new services or products, to achieve an automated zero-touch activation and to reduce the Total Cost of Ownership (TCO).

2.4.1 Workflows

A workflow is a definition of an executable process. The definition is at a detailed algorithmic level, suitable for control of interactions with activation targets. Workflows are composed from a set of workflow nodes, which are executed by the workflow manager, one node at a time.

Currently, there are several business processing steps that can be automated using HPSA workflow nodes, namely, extracting data from an incoming XML message, calculating derived parameters, requesting and updating data from external repositories such as an inventory database, sending messages to external processes, waiting for input from a human operator or external process and perform hardware or software configuration according to gathered parameters. Each workflow node is implemented as a Java class, extending a base node class with specializations for process nodes, rule nodes and switch nodes.

2.4.2 Plug-ins

A plug-in is a Java class that contains methods to perform configuration tasks related to a specific type of software or hardware component. Plug-ins provide at least an Atomic Task which is a java method that will be summoned from Activate nodes available in workflows. These atomic tasks are managed by a resource manager which is part of HPSA’s framework and which guarantees that every Atomic Task is executed atomically and reliably maintains the state of each transaction in order to recover transactions that were interrupted by a system failure.

The Atomic Tasks provided must be reversible and, for that purpose, the first parameter it receives is an operation parameter which takes either a DO_AND_CHECK or UNDO_AND_CHECK value used by the resource manager to manage the reversibility of a plug-in. Every task must return an ExecutionDescription, which is a java class provided by the resource manager library and indicates whether a plug-in executed successfully or not. If a plug-in fails its execution, either from internal causes or because it received an error from the managed device, the resource manager will call the provided Atomic Task with UNDO_AND_CHECK as the first parameter, indicating that it must reverse the managed device to its initial state.

Three generic plug-ins are supplied as part of the HP Service Activator core product: GenericCLI, GenericHTTP and LDAP plug-ins.

2.4.3 GenericCLI Plug-in

The CLI plug-in is controlled by a XML dialog control template. A dialog, for example, for a complex router device, may comprise a large number of commands, and so the control template may be quite large. The control document will contain information about how to establish a session with the device, a sequence of command-response exchanges, and finally how to terminate the session. The specification for each command response exchange includes the exact command line to be sent, the expected response and the patterns that allow recognition of possible error responses. Every dialog, even those with many commands, are executed as a single atomic task. In order to ensure clean failure of the task when an error response is received for a command, the CLI plug-in supports rollback of the command
sequence. Rollback command sequences use rollback commands that can be specified for each command-response exchange that is part of the sequence.

The XML dialog control template typically includes a number of strings whose values must be sourced from variables of the workflow node that invokes the atomic task, representing items such as username and password for authentication of the session/request by the target, names of devices, ports, interfaces and other objects that exist and must be manipulated on the target.

3 Architecture Design Requirements

NETCONF is a connection-oriented protocol, meaning that one and only one session should be opened for each device. Given the requirements defined by IETF for NETCONF and the requirements that are usually set for an HPSA plug-in, a set of requirements are also defined for this plug-in:

1) Support all operations defined in RFC 6241
2) SSH as transport layer
3) Validate content/inputs for all supported operations
4) Support a concurrent interaction with one or more network devices
5) It must be generic, meaning that it shouldn’t be vendor specific and HPSA or allow only a set of available configurations.
6) Provide an interface/schema to be used by HPSA workflows.
7) Plug-in Tasks should be atomic
8) Atomic tasks should be reversible
9) Atomic tasks must take an initial parameter indicating whether this is the DO_AND_CHECK or the UNDO_AND_CHECK invocation of the task

Even though SSH is mandatory in all NETCONF implementations, as seen above, there are already other transport protocol definitions for NETCONF and its implementation must be easily extensible to support the implementation of new transport protocols. The same goes for Operations, RFC 6241 defines a set of operations, yet, NETCONF servers may extend its capabilities and therefore, the client implementation should also be extensible enough to support new operations and capabilities.

4 GenericNETCONF

In order to address the requirements and to make it as extensible as possible, the GenericNETCONF structure is based on the one defined by RFC 6241 for NETCONF implementations, following a four-layer architecture with some adaptations to work as a plug-in for HPSA as illustrated in Figure 1.

![Figure 1. GenericNETCONF layers architecture](image)

In order to make the plug-in as generic as possible, the content layer must be defined outside the plug-in and received as input. This gives network managers and HPSA workflows the power to adapt the content that is sent to the device accordingly to any network or non-network related variables. Every plug-in must provide at least one Atomic Task which will the main interface from which the plug-in will be referenced. GenericNETCONF provides task Netconf Activate which receives the initial DO_AND_CHECK or UNDO_AND_CHECK parameter which is used by the resource manager to handle the plug-in reversibility. This task is also the interface used by HPSA to connect the content layer with the other existent layers.

The operations layer is divided in four main groups: Action, Rollback, Commit and Undo, al-
lowing the plug-in to support concurrent activations when used together with the provided Execution Modes. For each operation there is a validation of its content which is done before execution. The operations are implemented using a command design pattern, which makes extending an operations very easily. Each operation must implement two methods, the `validate()`, which receives the content for that operation and validates it, and the `execute()` method, which is the connection to the Messages and Transport layer to execute the operation on the device.

The `Messages` and `Transport` layers are managed by the open source implementation of JNC done by Tail-f. The library is used to manage the NETCONF sessions and to manage the capabilities’ verification. JNC is the most updated open-source implementation of a NETCONF client done in JAVA and its architecture allows easily the implementation of new capabilities on the plug-in. Also, even though it provides by default only SSH on the transport layer, it provides an interface to extend the transport protocols available. The library also encapsulates the operations in RPC envelopes before it sends it to devices.

`GenericNETCONF` provides serial and parallel execution modes, making it able to support a concurrent interaction with one or more network devices. The set of devices to be managed as well as the execution mode are received as an argument in the task. Four types of execution modes are available with GenericNETCONF:

1) **Serial (default):** operations for each device are executed independently in order
2) **Parallel:** Does not assure order in the configuration of devices. The communication with each device is run in parallel.
3) **Atomic_Serial:** Operations for each device run until either one fails or all of them finish. The plug-in will start Rollback operations if one operation fails, or Commit operations if every operation is finished with success.
4) **Atomic_Parallel:** The configuration for all devices is executed in Parallel mode, with the plug-in checking if all the configurations went successfully.

The execution is managed by the plugin’s `ExecutionManager`, it is responsible by executing all kind of operations for all devices and to manage the success or failure of its execution and notify HPSA of the success of its execution.

### 4.1 Evaluation

This work was tested using three Virtual Machines (VMs), created using VirtualBox. The first VM was configured with 2 GB of RAM and Red Hat 6.4 x64 as Operating System, the required to install HPSA [16]. Since the version used to test this plug-in was HPSA 6.1, the first machine runs Java™ SE Development Kit 6, Update 37 and uses Oracle Database Express Edition 11g for the database. Two other VMs were created with 512 MB of RAM, use also Red Hat 6.4 x64 as Operating System, and have `Netopeer` installed. `Netopeer` installation provides netopeer-cli, which works as a NETCONF client to manage devices through any terminal, and netopeer-server, which implements the server used to test GenericNETCONF. `Netopeer` Server has a multi-level architecture and can be extended with modules to control different areas of the managed device. By default, netopeer-server comes with three pre-installed modules: `ietf-interfaces` which implements RFC 7223 [17], `ietf-system` which implements RFC 7317 [18] and a `turing-machine` module which implements a dummy machine to test NETCONF implementations.

#### 4.1.1 Junit tests

Since most of this project is done in JAVA, a set of JUnit tests was created taking in consideration that GenericNETCONF can be possibly extended with custom operations and new operations which can be defined accordingly to the work done by IETF’s NETCONF working group. These tests were divided in four main groups:

1) **Input Tests:** Mostly negative tests with invalid inputs, taking in consideration the schema defined for the XML specification and which should not be considered as valid by the plugin.
2) Connection Test: Connection success and errors are tested under the four different execution modes. In non atomic execution modes, the execution finishes successfully if the plugin is able to connect to at least one device, yet it may or not be consistent.

3) Operations and Content Tests: All operations are tested with valid and invalid content. The goal is to assure that the operation validates the input as expected and returns an error if the content is invalid.

4) Execution Mode Tests: Four operations are executed for both devices using each execution mode: Lock, get-config, edit-config, unlock.

All these sets of tests, assess the plug-in end to end, meaning that each test defines an input XML specification which calls the plug-in by its atomic task and asserts on its ExecutionDescriptor response code and description.

4.1.2 HPSA Plug-in test

To test the plug-in within HPSA environment, a solution was created taking into consideration the tests’ environment. The goal of this test was to update an inventory instance existent on HPSA with devices configuration state, and update configuration of these devices, accessing HPSA from its built-in provided web interface.

A simple inventory entity was created on HPSA with information relative to each device modules state. The inventory instance contains the two beans, one relative to device information and one other with modules state information. These beans are created in XML and deployed using a provided tool called InventoryBuilder. To be able to update the inventory and the devices’ configuration, three workflows were used used to enable or disable the truing-machine module and to retrieve the current configuration and update the inventory. The tests ran successfully.

4.1.3 Generic_CLI comparison

Considering that GenericCLI is the most common used plug-in in HPSA for network configuration, it makes sense to compare both plug-ins executing a simple operation. To do it, GenericCLI java classes were imported to an Eclipse project and a benchmark class was created. Both plug-ins were executed to enable the turing-module from one of the netopeer test machines.

Netopeer stores its candidate configuration in an XML document file typically located in /usr/local/etc/netopeer/cfgnetopeer/datastore.xml, which allows us to use a tool named sed to replace the content of the 52th line, which refers to the state of turing-module.

A significant advantage of using NETCONF instead of CLI is that error handling becomes easier. Using CLI, error messages are vendor specific, for that purpose GenericLCI provides a tag <Error> which will run every reply against a regular expression (regex). Using GenericNETCONF, errors are handled by the plug-in and its description can be accessed through the variable sterr of the returned ExecutionDescriptor. For this specific case, the regex expression to find an error coming from sed is not very complex, yet for more complex activations, it is difficult and takes too much effort to design regular expressions for all different errors for all the different managed devices.

To evaluate the performance of both plug-ins, the time to execute the aforementioned operation was measured. Let’s consider $Y$ the random variable representing the running time of an HPSA plug-in. Since $Y$ follows a Poisson distribution with parameter $\lambda$ and we want to estimate the average running time, we just have to estimate $\lambda$. For that, we will use the sequence of average running times given by $\{X_n\}_{n=1}^{\infty}$ such that $X_n = \frac{1}{n} \sum_{i=1}^{n} Y_i$, where $Y_i$ are independent and have the same distribution as $Y$. Therefore:

$$E[X_n] = E[\frac{1}{n} \sum_{i=1}^{n} Y_i] = \frac{1}{n} \sum_{i=1}^{n} E[Y_i] = \frac{1}{n} \times n \lambda = \lambda$$

(1)

$$Var[X_n] = Var[\frac{1}{n} \sum_{i=1}^{n} Y_i] = \frac{1}{n^2} \sum_{i=1}^{n} Var[Y_i] = \frac{1}{n^2} \times n \lambda = \frac{\lambda}{n}$$

(2)

Moreover we conclude that the estimator is consistent since $\lim_{n \to \infty} \frac{\lambda}{n} = 0$ and so the method converges.
Using the central limit theorem, a 95% confidence interval for \( \lambda \) is \( \lambda \pm q_{\alpha} \sqrt{\frac{\lambda}{n}} \), where \( q_{\alpha} = 1.96 \) is the 97.5\(^{th}\) percentile of the Normal(0,1). Thus, the method works as shown in Algorithm 1:

**Algorithm 1:** Compute a 95% confidence interval for an HPSA plugin’s running time

**Result:** A 95% confidence interval for the plugin’s running time

begin
\( n \leftarrow 0 \)
while \( \sigma \geq 1 \) do
Run an operation for one device and compute the average running time \( X_n \)
Compute the standard error \( \sigma \leftarrow \sqrt{\frac{X_n}{n}} \)
\( n \leftarrow n + 1 \)
else
\( I = [x_n - 1.96 \times \sqrt{\frac{x_n}{n}}, x_n + 1.96 \times \sqrt{\frac{x_n}{n}}] \)
return \( I \)

The interval obtained for GenericCLI was \( I = [4897.04, 4900.96] \) after 2451 runs with an average running time of 4899 ms and the interval obtained for GenericNETCONF was \( I = [628.04, 631.96] \) after 316 runs and an average running time of 630 ms. Taking a closer look, we can conclude that GenericCLI has some delay creating the ssh connection when compared with GenericNETCONF. Also, because GenericCLI took more runs to converge than GenericNETCONF, we conclude that the second is less variable than the first.

Even though GenericNETCONF was significantly faster, it was not created to replace GenericCLI. Both protocols have advantages over each other and that should be taken in consideration depending on the usage. GenericNETCONF default error handling significantly decreases not only implementation errors but also the development time, as it is not necessary to confirm all the possible error messages for all requests. Also, given that NETCONF is mostly XML, it is very easy, either for human reading as for machine interpretation, which decreases implementation errors and development time on handling replies. GenericNETCONF is also useful when executing a configuration on more than one device, as it gives the possibility to commit the managed devices’s configurations only when it is guaranteed that all devices can be configured with success. GenericCLI on the other hand, has the significant advantages of supporting most of the operations that can be executed in a given device and also the fact that CLI is present in most network devices.

### 5 Conclusion

TMF’s eTom defines the business processes that must be implemented by any CSP, but in order to make these processes work efficiently, the right tools that are able to automate those processes and make them faster and more error-free are necessary. NETCONF is the most recent network management protocol standardized by IETF. One of the main reasons for the creation of NETCONF was to automate the network management processes.

Given that more than 50% of the services activation market is controlled by a few top suppliers, it will for sure be a differentiator factor if one of these solutions implements a newly created network management protocol, more focused on the CSPs needs. HPSA is completely integrated in a complete Fulfillment solution and is one of the main solutions used nowadays, making it a reliable product in terms of support and maintenance.

HPSA provides the possibility for a network manager to create an activation workflow. This activation workflows can be created through several workflow nodes, where each node represents a specific action, such as sending a configuration to a specific device, interpret replies sent by network devices or even evaluate those replies. HPSA workflows can interact with the existing plug-ins using predefined templates which have a set of variables that must be defined by the workflow.

GenericNETCONF is a new plug-in, which provides HPSA with a NETCONF southbound interface, as such, its architecture is designed
to address not only the requirements specified for the network management protocol but also the requirements established for a generic plug-in. Conceptually, NETCONF is partitioned into four layers: Content, Operations, Messages and Transport. This plug-in covers all the last three layers, with the content for the operations to be executed being received as an XML input. The XML may specify instructions for several devices, that are interpreted and validated by the plug-in, which generates the messages and communicates with the target devices. The plug-in is able to interact with one or more devices and supports more than one type of concurrent execution, which can also be defined in the input XML document.

GenericNETCONF was tested with success using a typical workflow, which is probably the most common environment from where it would be summoned. A small comparison was made between GenericNETCONF and GenericCLI. GenericNETCONF is not intended to replace GenericCLI. However, it can be easily concluded that both have some pros and cons, mainly in terms of error handling, development time and concurrency handling (where NETCONF shows its value and flexibility), and where CLI has the added value of being able to do practically any operation in any device. This project was also evaluated by the interoperability of the functions it supports and by the possibility of new added operations, still maintaining the interoperability of the base operations.

ACKNOWLEDGMENTS

I would also like to acknowledge my dissertation supervisors Prof. Rui Cruz from IST, and João Valentim from Hewlett-Packard, for their insight, support, encouragement and sharing of knowledge that has made this work possible.

REFERENCES


