

Developer Days

Automation



Automation of Robot/CXTA test cases

Mikael Tidemar
Customer Delivery Software Architect
2022-11-30



The bridge to possible



After manual verification comes automation



- Test case creation is generally not an automated process
 - Manual configuration capture
 - Manual test case definitions
 - Manual test case files writing
- Any service update will require at least a new configuration capture



How do you deal with...

- Variations
- Upgrades
- Verifying the configuration pushed is same in different environments
- Agile sprint development constantly changing the expected results of tests

Automation engineers spend a lot of time on manually updating existing test cases

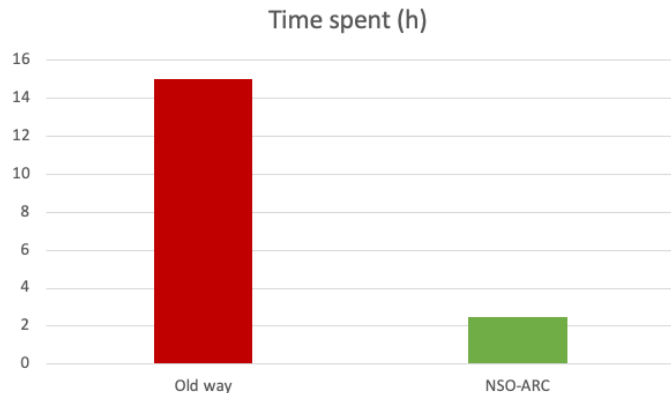
Customer case: L2&L3VPN

Cisco CX engineers delivered automated acceptance tests to a service provider

- 1 h meeting with developers
- 4 h develop first version of test, all variations
- 1 h to discuss packages
- 2 h change tests
- 1 h explain an issue and download fix
- 1 h to debug code and change test
- 2 h a new iteration with a new problem
- 3 h run in customer physical device lab, fix new issue and re-run

= 15 h for fixing a single use case!

From a very long (and real) 15 h down to an approximation of 2 h 17 min!



- 5 min configure service based on input from developers
- 10 minutes for creative variations
- 2 min run nso-arc action
- 10 min feedback to architect/developer
- 10 min download packages with fix, change the service, re-run nso-arc
- 10 min report an issue
- 10 min download packages with fix, re-run nso-arc
- 20 min new iteration with a new problem
- 1 h run in the customer physical devices lab, using nso-arc generated test cases is quick, most of the time spent interacting with customer

= 2 h 17 min

Introducing an automated QA approach for NSO projects

- Without additional time budgeted
- With (or without) a CI/CD pipeline setup
- Without learning Robot
- Without limited test planning in advance

Can it be done?



Automated Robot(file) Creation

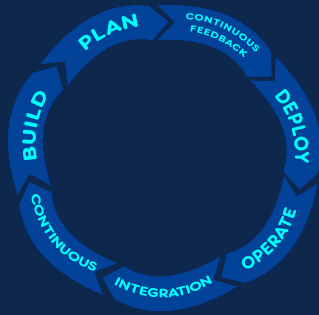
Using NSO and Robot framework



What do I get?

Configured service instances in CDB (**manually verified**) turned into automated test cases using Create/Read/Update/Delete in a matter of seconds

Simple Robot framework tests
Using well established standards (RESTCONF)



CI/CD Pipeline addressable NB interface

“Why not automate test case creation via pipeline?”

NSO-ARC is an action-package

Roughly ~2000 lines of code

Python files per action and helpers

Fast execution, scalable

Supports LSA, multi-layer stacked services and nano-services (plan)

Compatible with NSO 5.x / 6.x

Used by CX in delivery in multiple ongoing engagements

No RFM / kickers support

NSO-ARC is a unique tool

Unique in it's capturing capability

Works with live-status exec "show running-config" / rpc-request-shell-execute via NED

Gives you line-by-line config validation unfiltered by the NED configuration coverage

Diff comparison between before/after means similar results can be compared in lab/dev/pre-prod/prod

```
admin@ncs# arc-action create-robot all-service-instances output-path ARC-TESTS
```

Possible completions:

add-to-previous	Add test cases to latest nso_service_testing.robot file (if it exists in the chosen output path)
include-children	Include children of stacked services from testing (not recommended)
no-networking	Note: This option will not check actual device config changes
pre-config-cdb	List all xpaths for pre-configuration capture inside NSO CDB []
pre-config-devices	List devices for pre-configuration capture
test-in-isolation	Remove all services before testing each service instance in isolation (execution time is long!)

Simple input command options

Four main ways of interacting with the action:

arc-action create-robot **all-service-instances** **output-path** /tmp **no-networking**

test-in-isolation **include-children** **add-to-previous** **pre-config-cdb** [/xpath1 ...] **pre-config-devices** [device-name1 ...] (optional keywords & inputs)

arc-action create-robot **service-type** [/service-point1 /service-point2] **output-path** /tmp

arc-action create-robot **service-instance** [“/this-is-a-keypath{key1 key2}”] **output-path** /tmp

- All-service-instances gives generated robot file for every single service instance
- Service-type gives a generated robot file for all instances for the list of service-points given
- Service-instance gives generated robot file for the selected instances given as a keypath leaf-list
- output-path (mandatory input - press tab to view OS folders/files, use absolute or relative path)

Simple input command options

Modify-variation generator:

```
arc-action create-robot modify-variation { service-instance "/router-static-rfs:router-static-rfs{iosxr-cli0 test-vrf}"  
  parameter {  
    yang-name static-route-ipv4  
    positive-modify-values [ 1.2.3.0/29 2.3.4.5/29 ]  
    negative-modify-values [ test ]  
  }  
  parameter {  
    yang-name fw-ipv4  
    positive-modify-values [ 4.3.2.1 5.6.3.2 ]  
    negative-modify-values [ 311.111.11.1 424.111.2.3 ]  
  }  
} output-path NSO-ARC no-networking (optional)
```

What does it give as output?

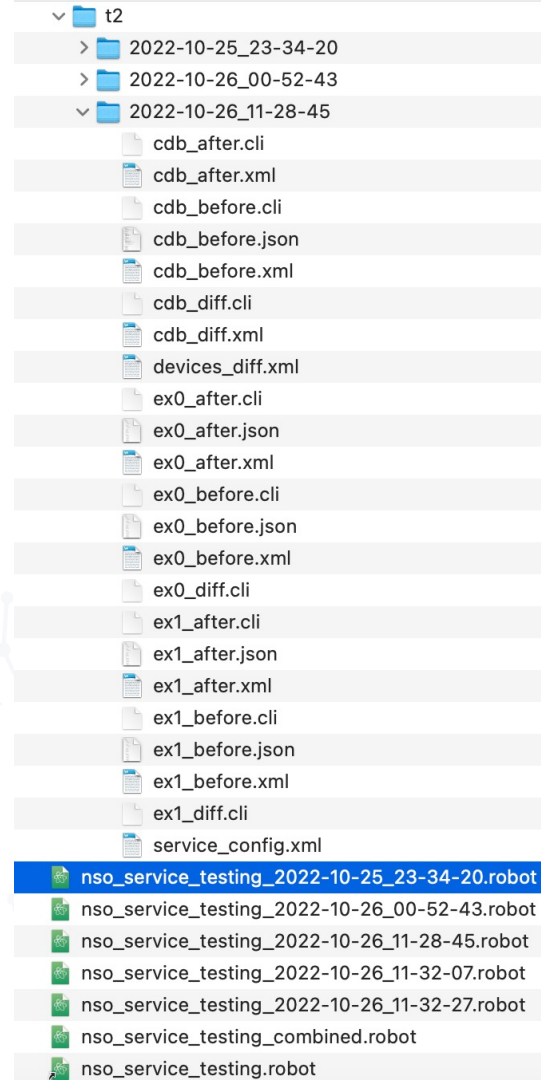
nso_service_testing_2022-10-25_23-34-20.robot

nso_service_testing.robot <- symlink to latest run

Robot test file generated,
a repeatable date and time stamped folder structure **per service instance** with the necessary files for validation

add-to-previous keyword allows for batch creation with individual options added into a nso_service_testing_combined.robot file

100% ready for implementation in a pipeline!



Demo

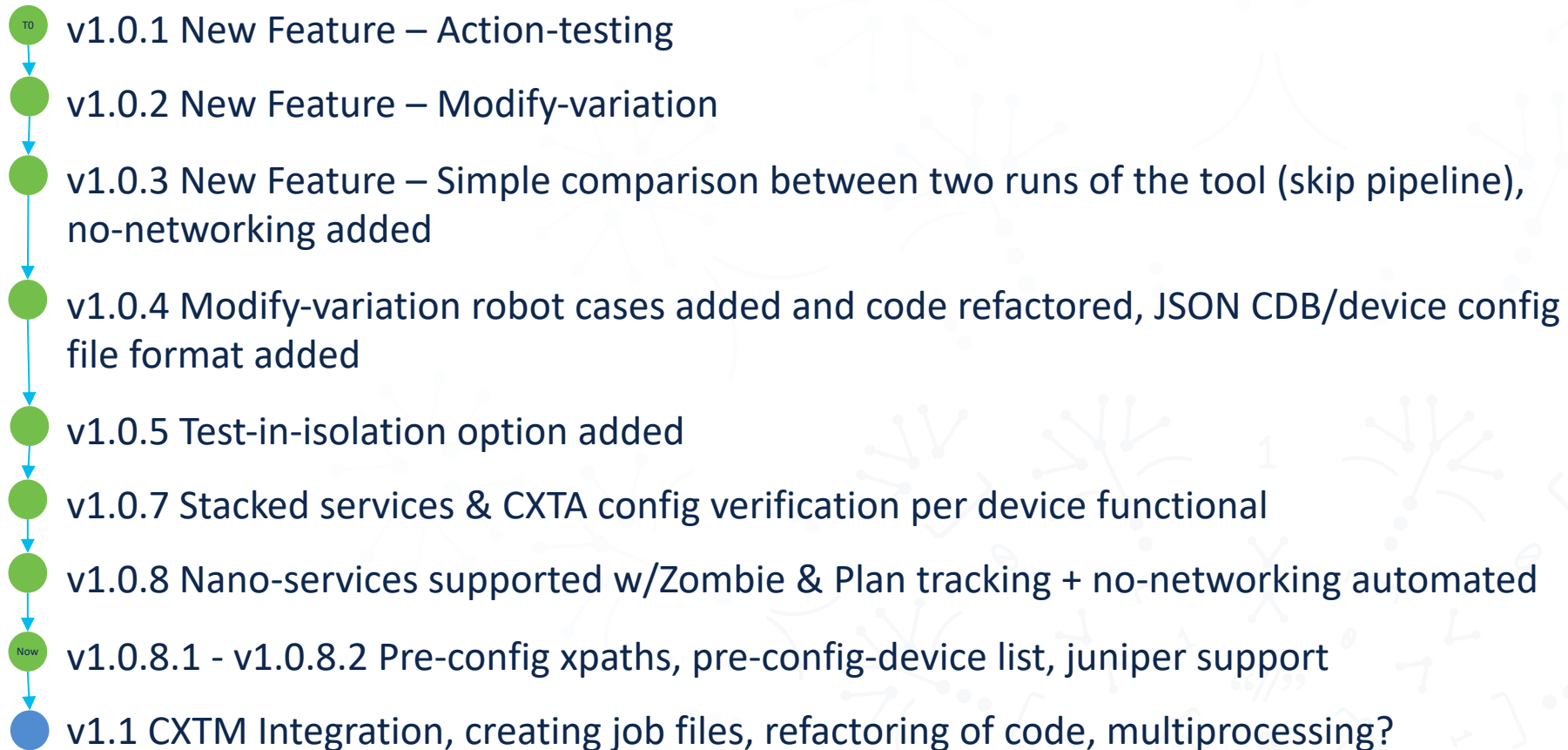


First customer PO: Swisscom

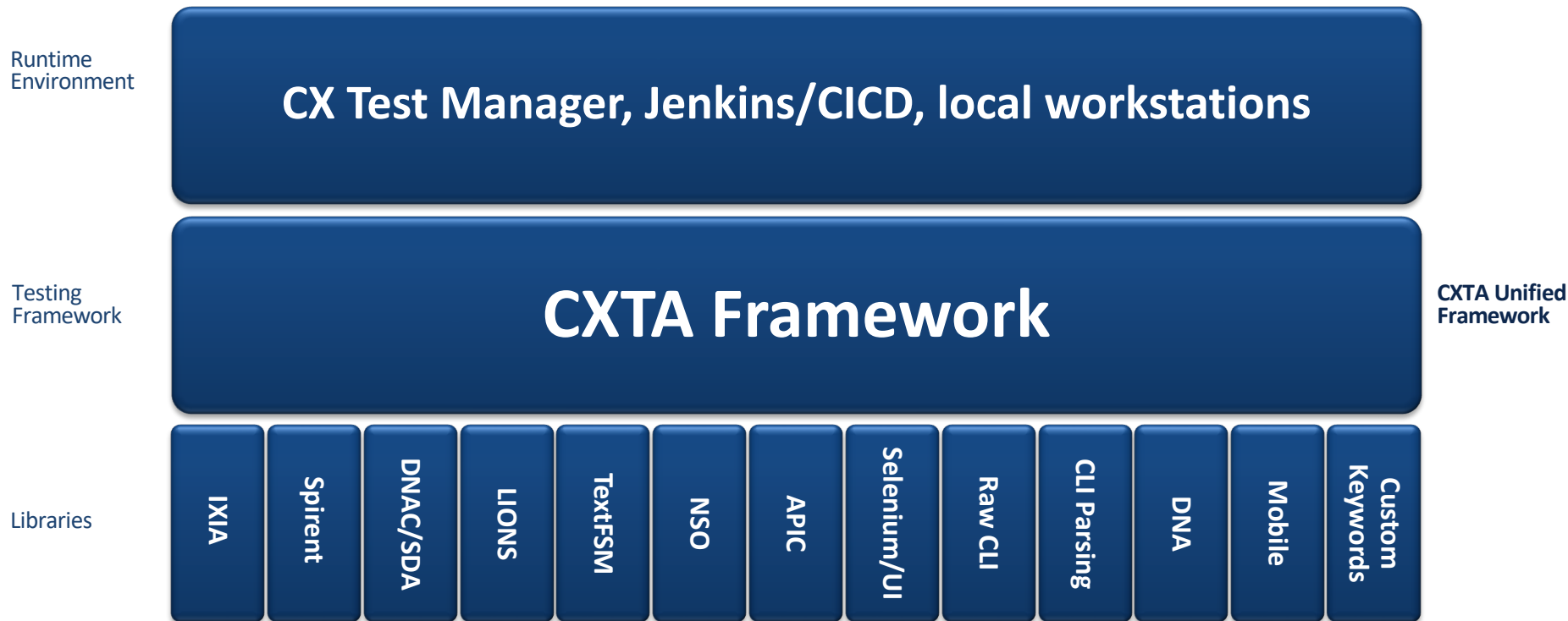
“This is what I’ve been doing manually”

“We want it now”

Timeline – Current Release v1.0.8.2



CXTA – CX Test Automation Framework





Why Robot?

- **Open source** Test Automation Framework which handles test abstraction using natural language keywords
- Large Ecosystem with a variety of **Keyword Libraries** already available
- Allows **test cases** to be written/controlled by **non-programmers** as most tests don't require complex scripting
- Easily integrates existing **Python** test **libraries** created within Cisco (PyATS) so we can re-use/leverage existing work

*Don't drown in test case creation
work, use our **ARC***

nso-arc-support@cisco.com



The bridge to possible