



What's up NSO?

Optimization and Troubleshooting

Sebastian Strollo
Principal Engineer
December 5, 2019

Introduction



Let's start with a demo

- One NSO instance
- Two devices, ios0 and ios1
- A (nonsenical) template based service that configures interfaces

This is what I will do

```
edit ifc demo
set GigabitEthernet 0/10 address 10.10.10.1
set device [ ios0 ios1 ]
top
show | compare
commit dry-run outformat native
commit
```

Huh?
Why did that take
so long? What is
going on?

- Let's look at the logs...

Which log? The major players:

Log	Where	What...
audit.log	ncs.conf	...was done in a n/b interface (cli)
audit-network.log	ncs.conf	...did NSO send to a device
devel.log	ncs.conf (levels!)	...is going on inside NSO
ncs.log	ncs.conf	...is going on with NSO high level
ncs-java-vm.log	setting / enable per class	...did the Java code do
ncs-python-vm.log	setting / typically one per package	...did the Python code do
ned-NED-DEVICE.trace	device setting	...is going on in the NED / device
netconf.log	ncs.conf	...was sent to NSO via NETCONF

Let's look at the audit.log

```
<INFO> TS HOST ncs[PID]: audit user: admin/50 CLI 'edit ifc demo'
<INFO> TS HOST ncs[PID]: audit user: admin/50 CLI done
<INFO> TS HOST ncs[PID]: audit user: admin/50 CLI 'set GigabitEthernet 0/10 address 10.10.10.1'
<INFO> TS HOST ncs[PID]: audit user: admin/50 CLI done
<INFO> TS HOST ncs[PID]: audit user: admin/50 CLI 'set device [ ios0 ios1 ]'
<INFO> TS HOST ncs[PID]: audit user: admin/50 CLI done
<INFO> TS HOST ncs[PID]: audit user: admin/50 CLI 'show'
<INFO> TS HOST ncs[PID]: audit user: admin/50 CLI done
<INFO> TS HOST ncs[PID]: audit user: admin/50 CLI 'commit dry-run outformat native'
<INFO> TS HOST ncs[PID]: audit user: admin/50 CLI done
<INFO> TS HOST ncs[PID]: audit user: admin/50 CLI 'commit'
```

Where **TS** is something like **5-Dec-2019::08:45:26.962**

The two fundamental activities in NSO

- USER SESSION
- TRANSACTION

The USER SESSION

- Belongs to an authenticated user
- Is identified by a single integer
- Is created when a n/b API authenticates (e.g. user logs into the CLI)
- The user session also has:
 - A “context” (name of the n/b interface or user definable when coming from programmatic api)
 - The special context “system” is an unauthenticated context only allowed from programmatic api on localhost
 - A remote endpoint address (i.e. IP address when logging in remotely)
- Is the parent of transactions
- (Is sometimes referred to as “session id”, “usid”, or “usess” in logs)

The TRANSACTION

- Always belongs to a USER SESSION (the nacm rules for that session determines what that transaction can do)
- Is identified by a single integer
- Is used to read and *write* towards NSO (CDB)
- Typically started implicitly by n/b API (but explicitly when using the programmatic APIs)
- (Is sometimes called “tid” or “thandle” in logs)

Now that we know
a bit more about...

...User Sessions and
Transactions...

...let's look at the audit-network.log

```
<INFO> 5-Dec-2019::08:47:01.807 HOST ncs[PID]: audit_network
      user: admin/50 thandle 1519 hostname HOST device ios0
BEGIN EDIT
interface GigabitEthernet0/10
  no shutdown
  no switchport
  ip address 10.10.10.1 255.255.255.0
exit
END EDIT
```

```
<INFO> 3-Dec-2019::08:47:01.807 HOST ncs[PID]: audit_network
      user: admin/50 thandle 1519 hostname HOST device ios1
BEGIN EDIT
interface GigabitEthernet0/10
  no shutdown
  no switchport
  ip address 10.10.10.1 255.255.255.0
exit
END EDIT
```

But hang on...

...remember what we
were doing?

Let's get back on track, maybe the devel.log?

```
...
<DEBUG> 5-Dec-2019::08:46:45.897 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: ncs: service /ifc[name='demo']:
create: service create ok
<DEBUG> 5-Dec-2019::08:46:45.917 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: ncs: service /ifc[name='demo']:
post-modification: service post-modification...
<DEBUG> 5-Dec-2019::08:46:45.918 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: ncs: service /ifc[name='demo']:
post-modification: service post-modification ok
<DEBUG> 5-Dec-2019::08:46:45.927 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: run transforms and transaction hooks done
<DEBUG> 5-Dec-2019::08:46:45.927 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: mark inactive...
<DEBUG> 5-Dec-2019::08:46:45.930 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: mark inactive done
<DEBUG> 5-Dec-2019::08:46:45.930 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: pre validate...
<DEBUG> 5-Dec-2019::08:46:45.934 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: pre validate done
<DEBUG> 5-Dec-2019::08:46:45.934 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: run validation over the changeset...
<DEBUG> 5-Dec-2019::08:46:45.943 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: run validation over the changeset done
<DEBUG> 5-Dec-2019::08:46:45.943 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: run dependency-triggered validation...
<DEBUG> 5-Dec-2019::08:46:45.943 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: run dependency-triggered validation done
<DEBUG> 5-Dec-2019::08:46:45.943 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: check configuration policies...
<DEBUG> 5-Dec-2019::08:46:45.943 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: check configuration policies done
<DEBUG> 5-Dec-2019::08:46:45.943 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: leaving validate phase for running
<DEBUG> 5-Dec-2019::08:46:45.944 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: entering write-start phase for running...
<DEBUG> 5-Dec-2019::08:46:45.944 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: cdb: write-start
<DEBUG> 5-Dec-2019::08:46:45.944 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: check data kickers...
<DEBUG> 5-Dec-2019::08:46:45.944 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: check data kickers done
<DEBUG> 5-Dec-2019::08:46:45.949 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: leaving write-start phase for running
<DEBUG> 5-Dec-2019::08:46:45.949 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: entering prepare phase for running...
<DEBUG> 5-Dec-2019::08:46:45.949 HOST ncs[101]: ncs commit progress db=running usid=50 thandle=1519: cdb: prepare
...
```

Clearly useful...

- But sometimes it feels like looking for the needle in a haystack

Progress Trace – *available since NSO 4.6*

- Extension of CLI feature: commit | details
- Resulting reports can be saved to:
 - File (in CSV or log format, most efficient)
 - Oper Data (easy to access, remember to set limit)
- Verbosity level controls how much output
- Filter controls source
- *Remember to enable in ncs.conf (and add hide-group to access cli)*

Progress Trace – Verbosity Levels

- *normal* – Phases and steps of a transaction/action (*default*)
- *verbose* – Service and device phases (including durations)
 - Useful to get an overview over where time is spent in a transaction/action.
- *very-verbose* – Details of service, device and internal operations
- *debug* – More details plus XPath trace
 - Usually too much info, but can be useful for debugging

Progress Trace – Filters

- *all-devices* – only device events
- *all-services* – only service events
- *context* – events originating from specified context
 - Example: cli, netconf, restconf, webui, system, ...
- *device* – events for specified device(s)
- *device-group* – events for specified group
- *local-user* – events for the specified local user
- *service-type* – events for the specified service type

Progress Trace - Headings

- Each trace entry always include:
 - Timestamp, Transaction ID, User Session ID
- And when applicable:
 - Context, Subsystem, Phase (validate, write-start, prepare, commit, abort)
 - Service, Service Phase
 - Commit Queue ID
 - Node name, Device Name, Device Phase
 - Package
 - Duration
 - Message

Progress Trace

DEMO

Observability



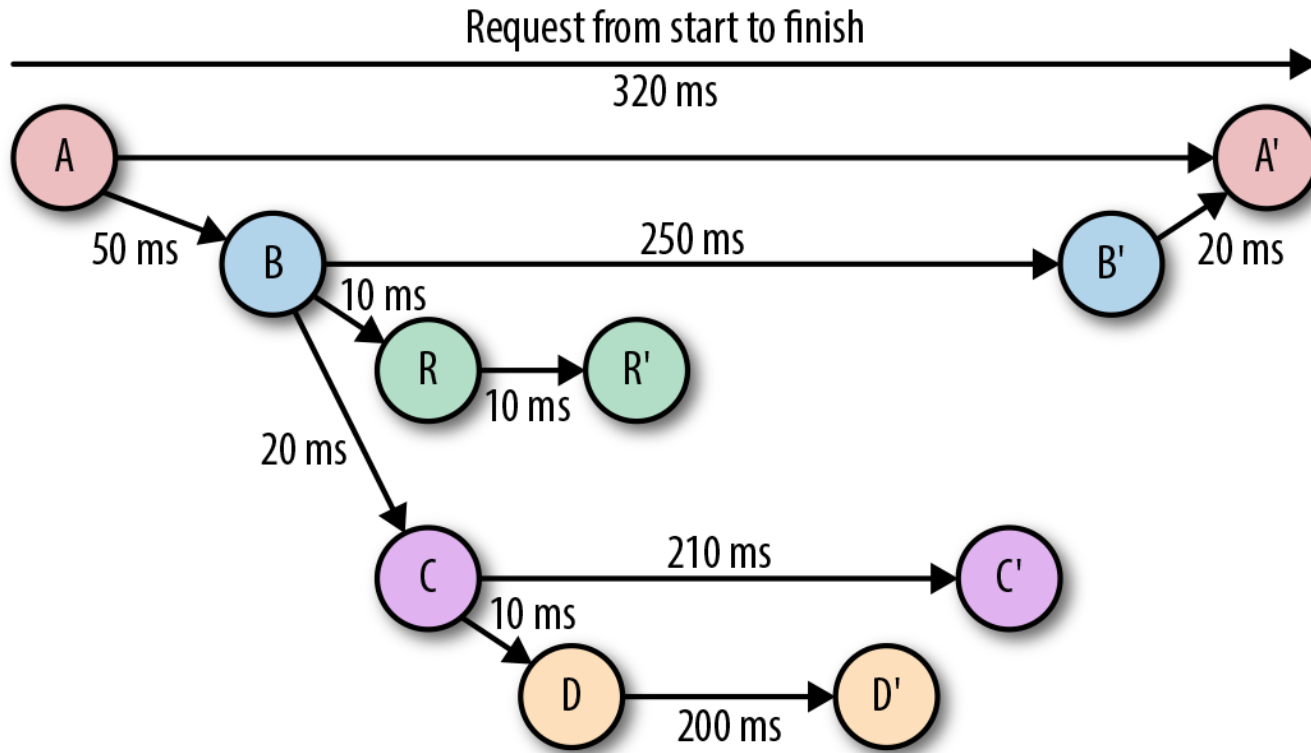
The Three Pillars of Observability

- (Event) Logs
 - An immutable, timestamped record of discrete events that happened over time
- Metrics
 - Numeric representation of data measured over intervals of time
- Traces

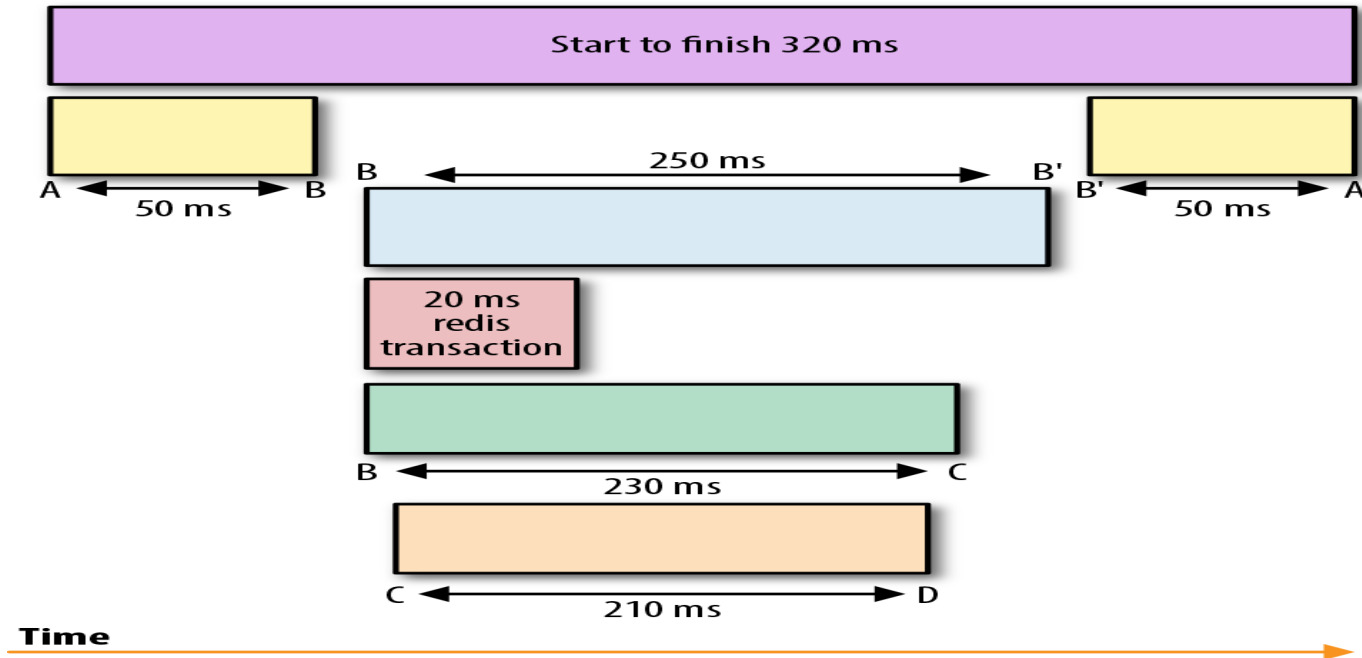
What defines a trace

- “A trace is a representation of a series of causally related distributed events that encode the end-to-end request flow through a distributed system.”
- Each trace has a unique ID
- A trace consists of “spans”, each span representing a specific activity
- A trace can typically be arranged in a directed acyclic graph

Like so:



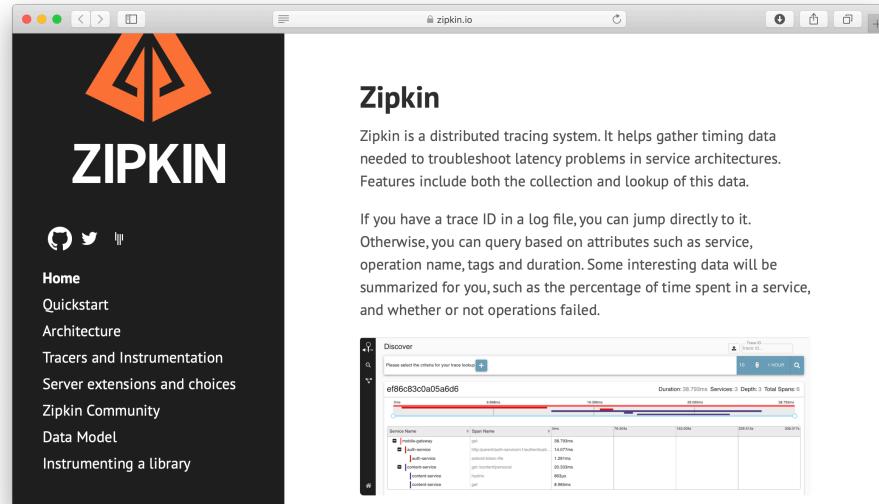
And even more useful, visualizing spans



Traces, continued

- Currently has a lot of attention (due to complex microservice applications being built)
- Open Source tools to visualize traces: Jaeger and Zipkin
- Con: Hard to retrofit into existing systems

Prototyping with Zipkin



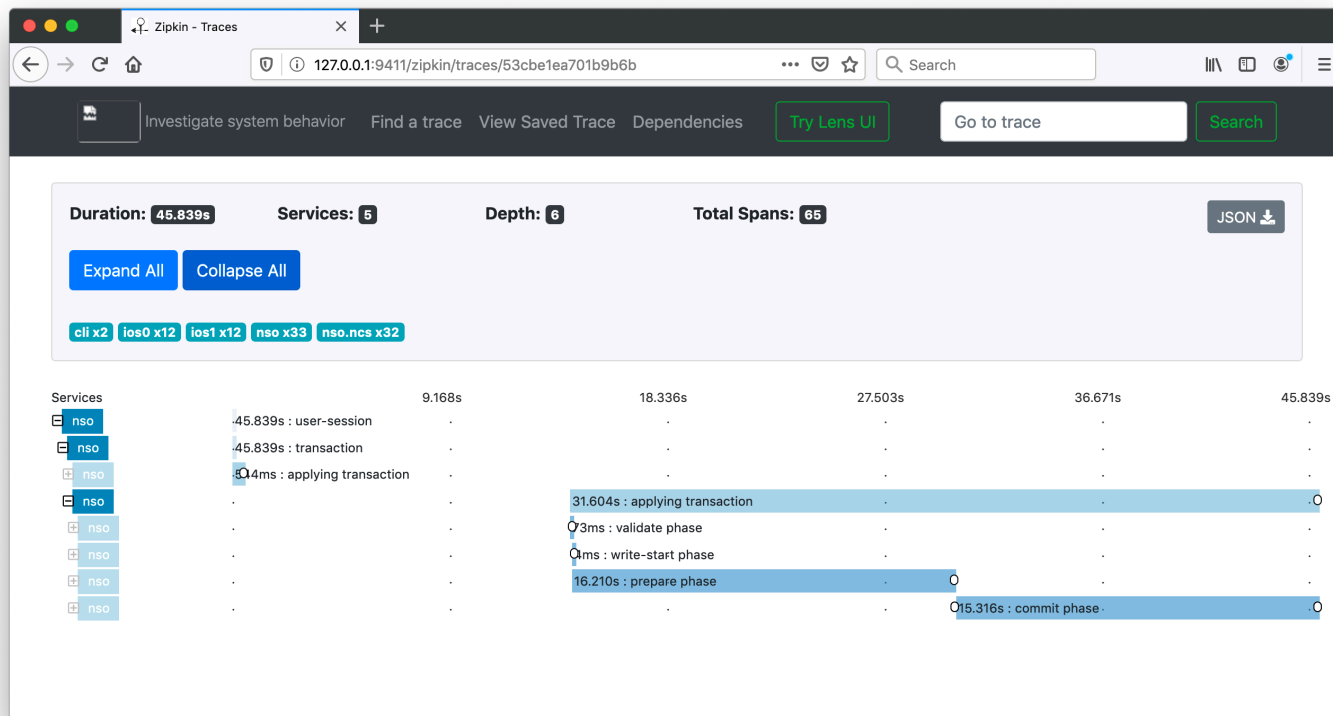
Prototyping with Zipkin, purpose:

- Learn more about tracing
- Share a usable tool (up on Github next week)
- Use as requirement on future NSO feature

But hang on...

...remember what we
were doing?

Right, that slow commit! Let's have a look...



Prototyping with Zipkin, next steps

- Short term:
 - Publish script on GitHub (<https://github.com/NSO-developer>)
 - Sort out what endPoints mean in the NSO context
 - LSA tracing (kind of works already)
- Medium term:
 - “Continuous export” (i.e. not batching from a file)
- Longer term:
 - Built in trace support into NSO (opentracing.io?)
 - Extend trace support to NEDs
 - Extend trace support to service applications



The phases of a (simple) transaction



The phases of a transaction



1. Run any pre-lock code
2. **Take transaction lock**
3. Run transforms and hooks (including any service code)
 - Amends transaction with further changes
4. Validate
 - According to YANG statements (and external validation)

The phases of a transaction



- Send changes to CDB

The phases of a transaction



1. CDB writes changes to disk
2. NSO device manager calculates which devices are involved (if any) and what to send to them (“southbound diff”)
 - **Device Locks are checked here (if taken transaction will fail)**
3. NSO device manager sends CLI commands to CLI NED:s (and edit-config to NETCONF NED:s etc)

The phases of a transaction



1. CDB writes transaction mark to disk
2. Kickers and Subscribers are notified of the changes
3. Device manager tells devices to “persist” change
4. Device manager tells NED:s to update transaction id
5. **Release transaction lock**

Transaction Lock

- Single lock per NSO node
- Taken when a write transaction is being *applied*
- Taken regardless of if the transaction involves device communication or not
- In standard mode the “southbound diff calculation” and “device communication” is done while holding the lock
- When using the commit queue the lock is released *before* “southbound diff calculation”

Device Lock

- One exclusive lock per device
- Taken for the duration of a sync operation
- *Note:* if the sync-from operations results in any changes that need to be written then the transaction lock is needed to write these changes

Commit Queue and Transaction Phases

- When using the commit queue, the validate and write-start are the same
- All device communication *and* “southbound diff” calculation are taken out of the remaining two steps
- When the transaction is completed the commit queue runs independently and will update the devices when they are available

The phases of a transaction – commit queue



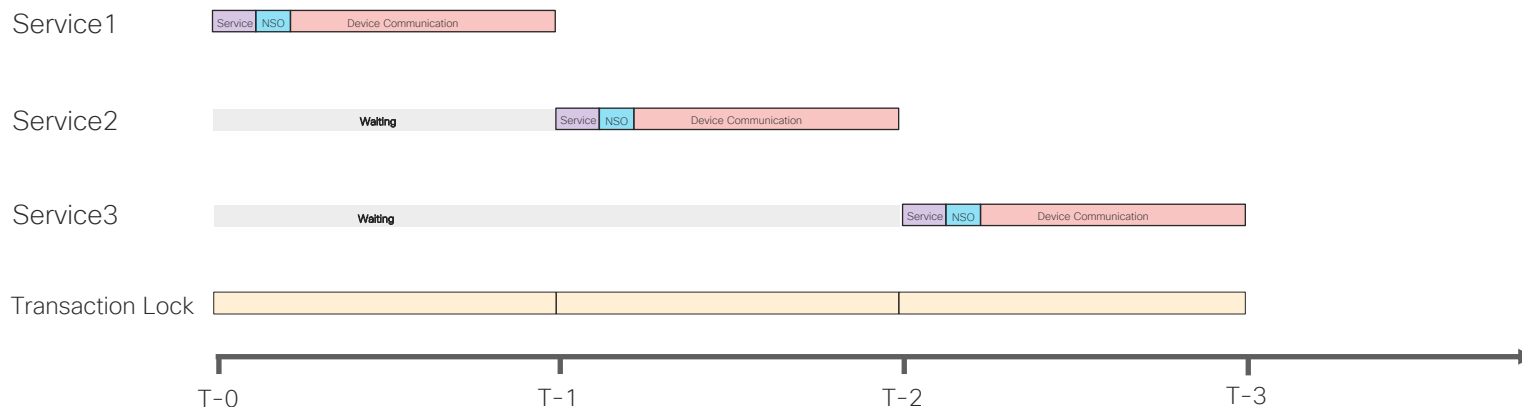
1. CDB writes changes to disk
2. NSO device manager calculates which devices are involved (if any) ~~and what to send to them (“southbound diff”)~~
 - **Device Locks are checked here (if taken transaction will fail)**
3. ~~NSO device manager sends CLI commands to CLI NED:s (and edit config to NETCONF NED:s etc)~~

The phases of a transaction – commit queue



1. CDB writes transaction mark to disk
2. Kickers and Subscribers are notified of the changes
- ~~3. Device manager tells devices to “persist” change~~
- ~~4. Device manager tells NED:s to update transaction id~~
5. Add transaction info (device list, and changes) to commit queue
6. Release transaction lock

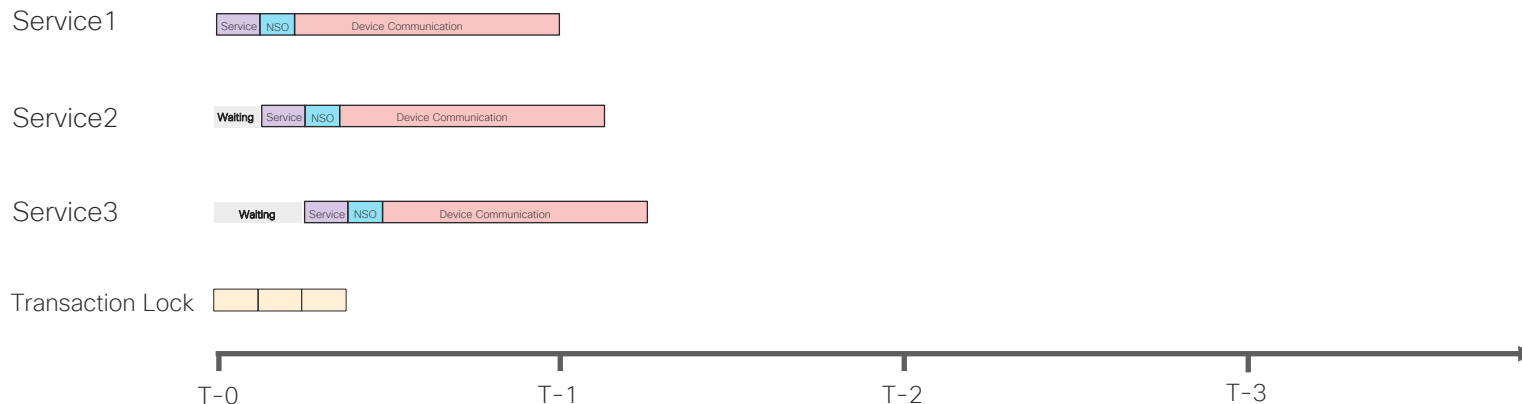
Transaction Time Line – Standard Mode



- This example shows three separate service intents entering NSO at roughly the same time.
- Since NSO is running in Standard Mode, transactions are serialized through the system and all three parts (service code, NSO protocol calculation and device communication) are inside the transaction lock.
- This has the effect that Service2 will have to wait until Service1 is fully provisioned, and service3 will wait until both Service1 and Service2 are done.

*This example does not show an accurate representation of the time spent in each phase of the transaction. It is rather focused on readability. Typically the *Service* and *NSO Protocol calculation* parts are in the (ms) time frame and *Device Communication* part is in the multiple seconds time frame.

Transaction Time Line – Commit Queues



- This example shows three separate service intents entering NSO at roughly the same time.
- Since NSO is running with Commit Queues, transactions are serialized through the short lived service code part, typically in the *ms* time frame. The NSO transaction lock is only active during the *Service Code* part.
- This has the effect that Service2 will have to wait until the *Service Code* part of Service1 is done, and service3 will wait until the *Service Code* parts of Service1 and Service2 are done. The *NSO Protocol Calculation* and *Device Communication* parts of all service intents/transactions are done in parallel.

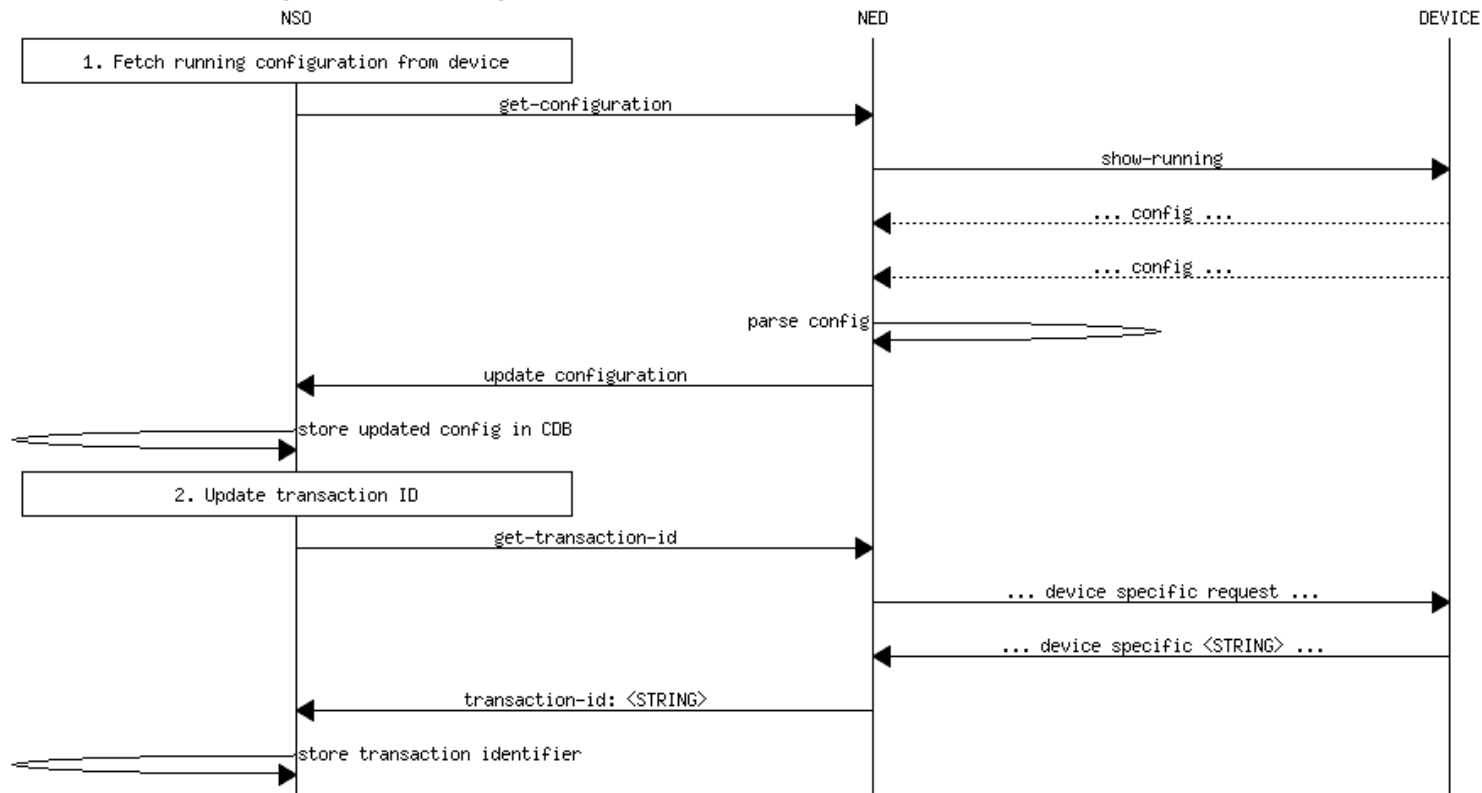
Understanding Sync



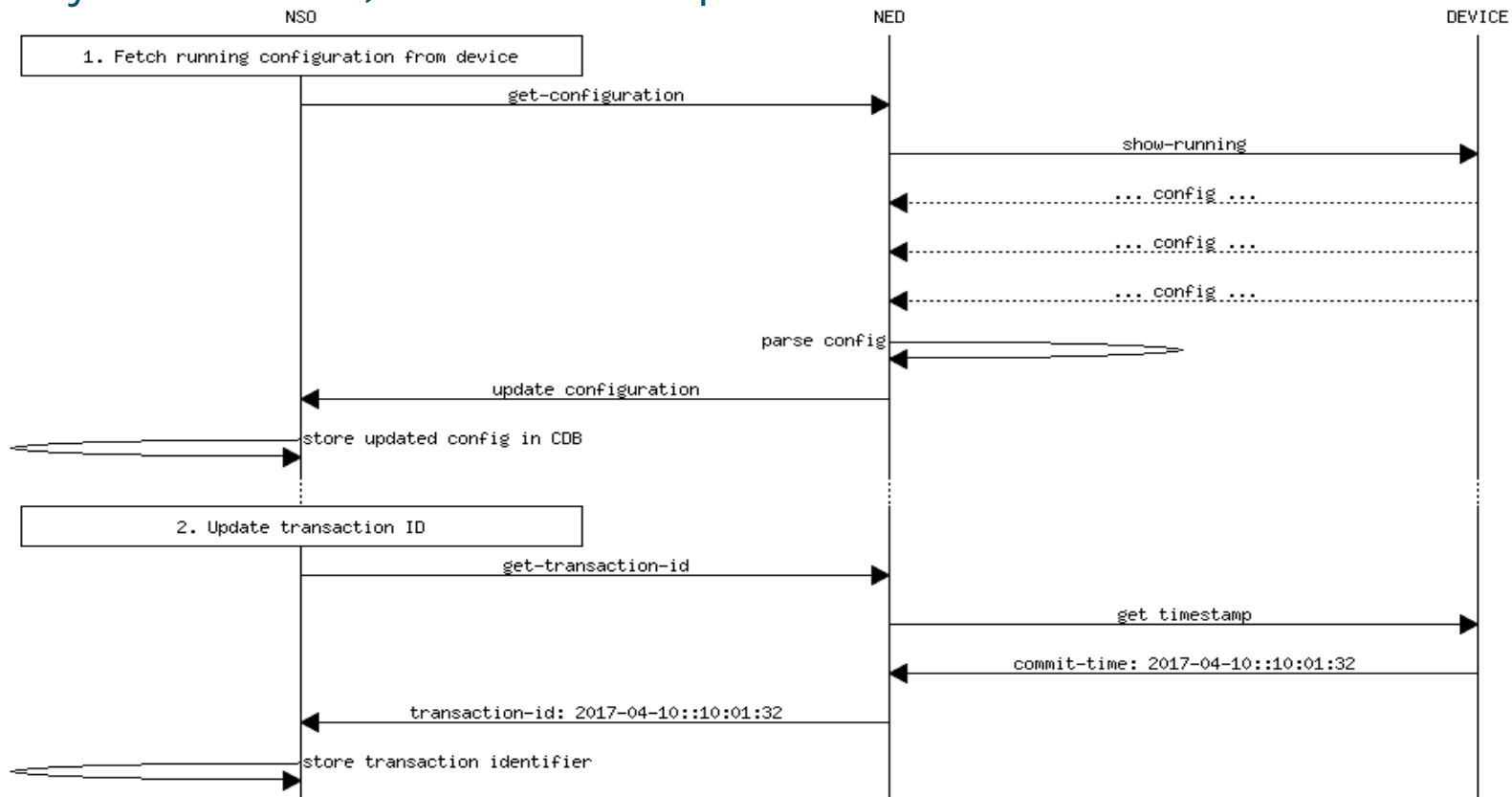
How does NSO know if a device is in sync?

- NSO stores a *transaction identifier* for every device
- The trans-id is updated every time a transaction is performed towards that device and on sync-from
- The trans-id is opaque to NSO – it is up to the NED to produce it
- Some devices have native support (e.g. a commit timestamp)
- *Default for a CLI NED is to compute a checksum of “show running”*
- Many NED's have specific ned-settings to use a different algorithm

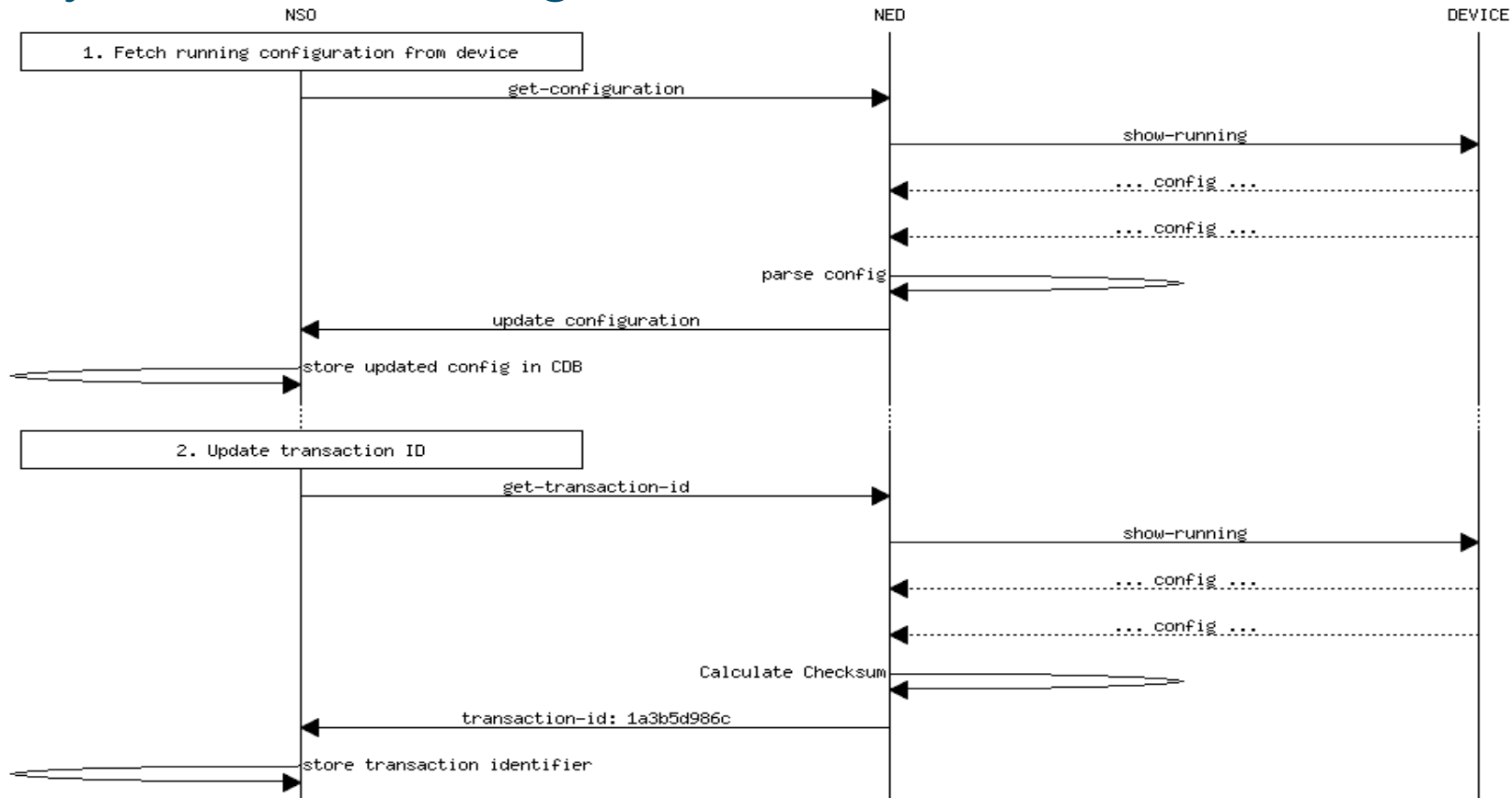
Anatomy of “sync-from”



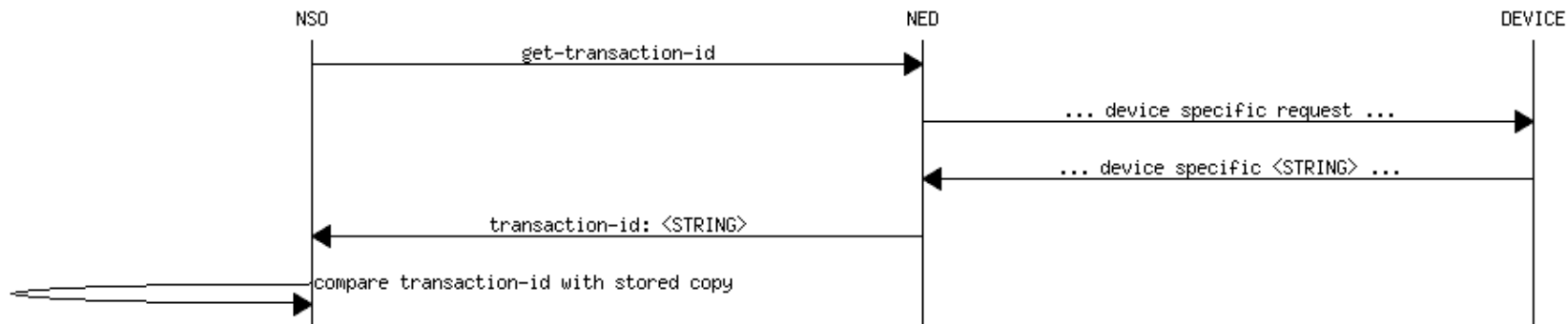
sync-from, timestamp as transaction identifier



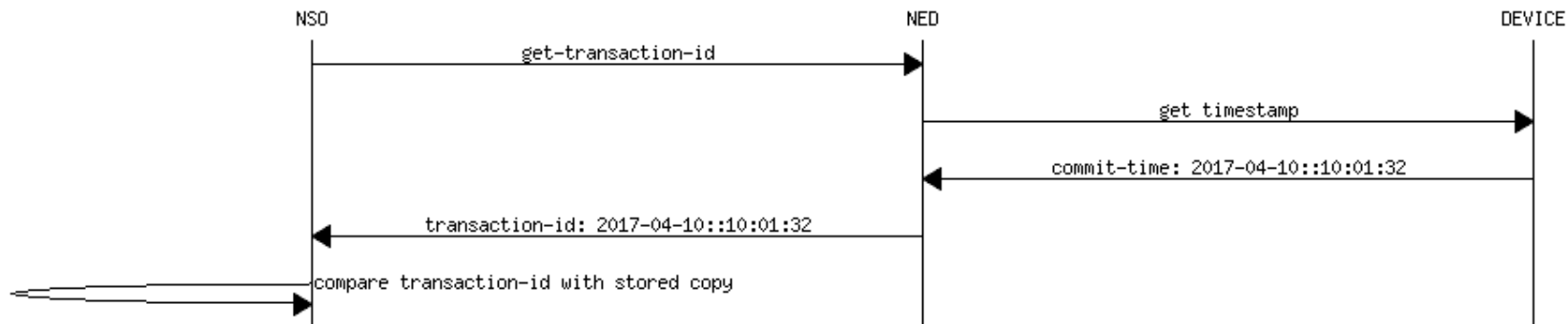
sync-from, config-hash as transaction identifier



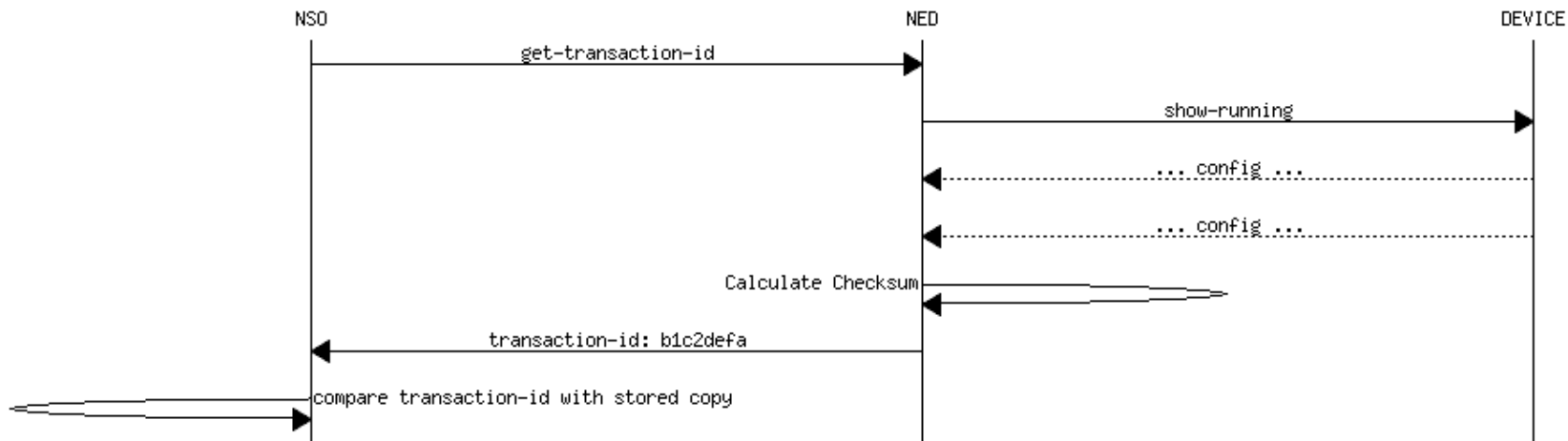
Anatomy of “check-sync”



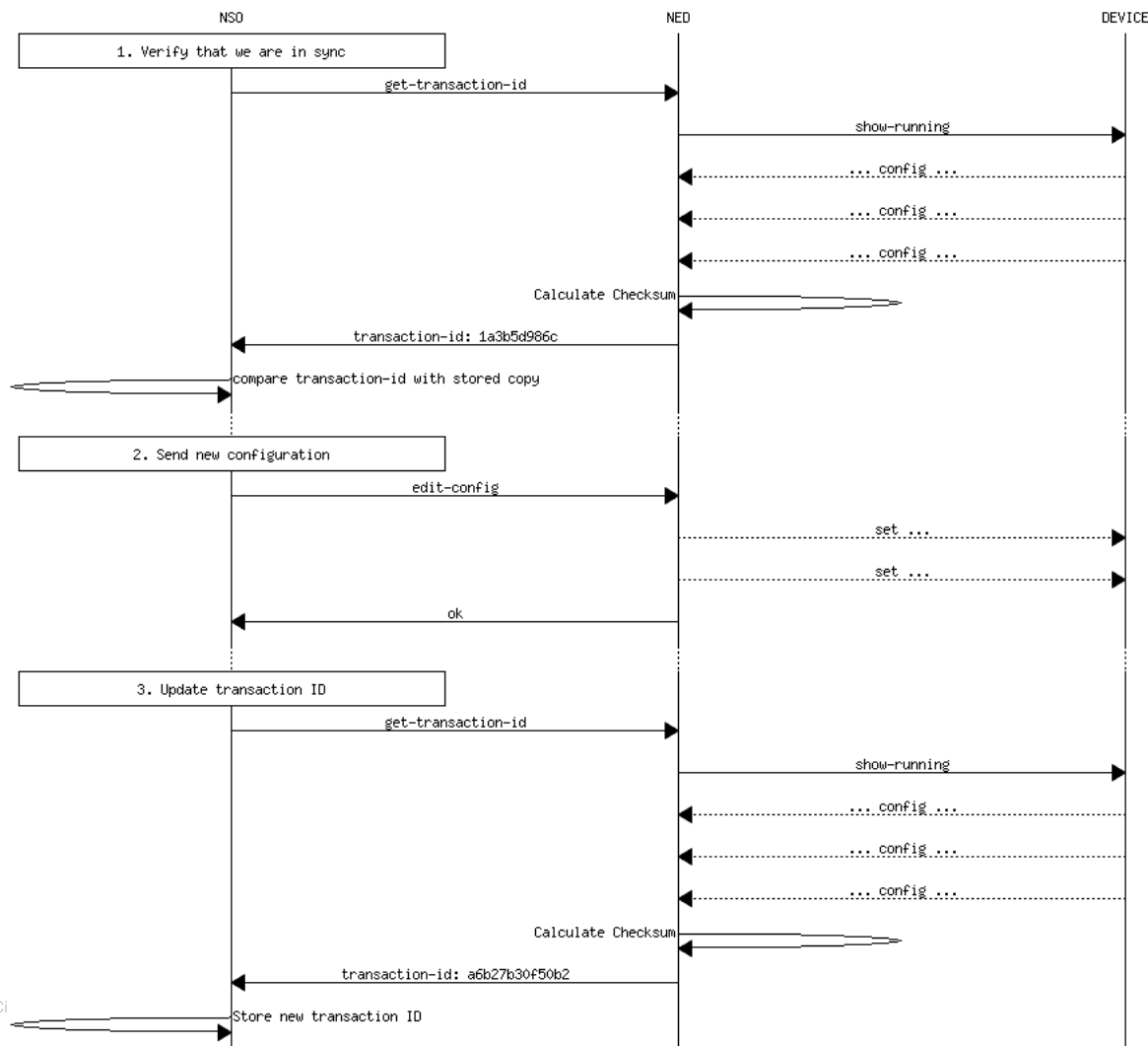
check-sync, timestamp as transaction identifier



check-sync, config-hash as transaction identifier



Regular commit sequence



Demo YANG models



```
module tag-value {  
    namespace "http://com/example/tag-value";  
    prefix tv;  
  
    container tags {  
        list tag {  
            key name;  
            leaf name {  
                type string;  
            }  
            leaf value {  
                type string;  
            }  
        }  
    }  
}
```

```

module ifc {
    namespace "http://com/example/ifc";
    prefix ifc;

    import ietf-inet-types {
        prefix inet;
    }
    import tailf-common {
        prefix tailf;
    }
    import tailf-ncs {
        prefix ncs;
    }

    list ifc {
        key name;

        uses ncs:service-data;
        ncs:servicepoint "ifc";

        leaf name {
            type string;
        }
    }
}

leaf-list device {
    type string;
    tailf:non-strict-leafref {
        path "/ncs:devices/ncs:device/ncs:name";
    }
}

choice interface {
    leaf FastEthernet {
        type string {
            pattern "[0-9]+/[0-9]+";
        }
    }
    leaf GigabitEthernet {
        type string {
            pattern "[0-9]+/[0-9]+";
        }
    }
}

leaf address {
    type inet:ipv4-address;
}

leaf dns {
    type string;
}
}
}

```



```

<config-template xmlns="http://tail-f.com/ns/config/1.0"
                  servicepoint="ifc">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>{/device}</name>
      <config>
        <interface xmlns="urn:ios">

          <FastEthernet>
            <name>{/FastEthernet</name>
            <ip> <dns><view-group>{/dns}</view-group></dns>
              <address>
                <primary>
                  <address>{/address}</address>
                  <mask>255.255.255.0</mask>
                </primary>
              </address>
            </ip>
          </FastEthernet>
          <GigabitEthernet>
            <name>{/GigabitEthernet</name>
            <ip>
              <address>
                <primary>
                  <address>{/address}</address>
                  <mask>255.255.255.0</mask>
                </primary>
              </address>
            </ip>
          </GigabitEthernet>

        </interface>
      </config>
    </device>
  </devices>
</config-template>

```