New Relic®

Telemetry
Swiss Army Knife in 10 minutes

Vladimir Ulogov

October 22, 2020

Safe Harbor

This presentation and the information herein (including any information that may be incorporated by reference) is provided for informational purposes only and should not be construed as an offer, commitment, promise or obligation on behalf of New Relic, Inc. (“New Relic”) to sell securities or deliver any product, material, code, functionality, or other feature. Any information provided hereby is proprietary to New Relic and may not be replicated or disclosed without New Relic’s express written permission.

Such information may contain forward-looking statements within the meaning of federal securities laws. Any statement that is not a historical fact or refers to expectations, projections, future plans, objectives, estimates, goals, or other characterizations of future events is a forward-looking statement. These forward-looking statements can often be identified as such because the context of the statement will include words such as “believes,” “anticipates,” “expects” or words of similar import.

Actual results may differ materially from those expressed in these forward-looking statements, which speak only as of the date hereof, and are subject to change at any time without notice. Existing and prospective investors, customers and other third parties transacting business with New Relic are cautioned not to place undue reliance on this forward-looking information. The achievement or success of the matters covered by such forward-looking statements are based on New Relic’s current assumptions, expectations, and beliefs and are subject to substantial risks, uncertainties, assumptions, and changes in circumstances that may cause the actual results, performance, or achievements to differ materially from those expressed or implied in any forward-looking statement. Further information on factors that could affect such forward-looking statements is included in the filings New Relic makes with the SEC from time to time. Copies of these documents may be obtained by visiting New Relic’s Investor Relations website at ir.newrelic.com or the SEC’s website at www.sec.gov.

New Relic assumes no obligation and does not intend to update these forward-looking statements, except as required by law. New Relic makes no warranties, expressed or implied, in this presentation or otherwise, with respect to the information provided.

What is this ?

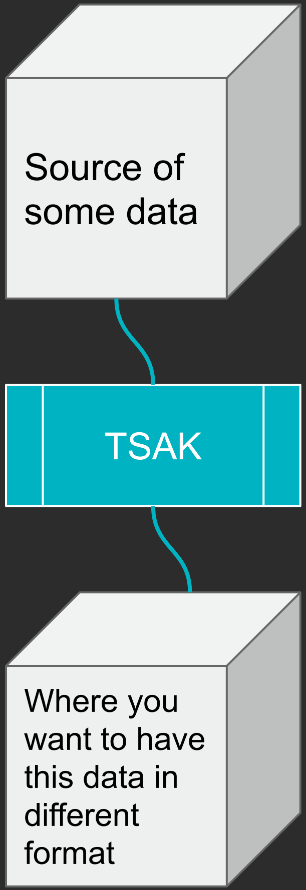
“Telemetry Swiss Army Knife” (or TSAK, for shorts) is a set of tools designed to deal with specific problem:

Accept or acquire a stream of data containing metrics, translate them to the format that we expecting on another end and provide it for the feeding.

What is this ?

Is that some kind of converter ?

Yes, this is a programmatic tool, that somehow acquired data from the source, converted them to the format, expected on destination and delivering it.



Source of some data

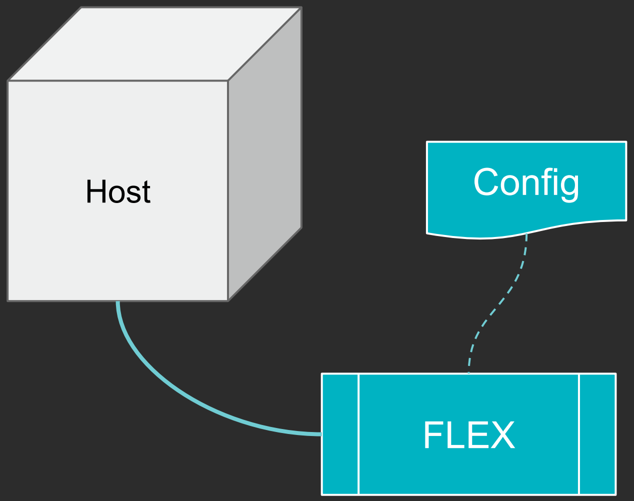
TSAK

Where you want to have this data in different format

What is this ?

But we do have an OHI and a Flex ?

FLEX and TSAK are quite different in what they are designed to do. While FLEX is a part of OHI (On-Host Integration) and designed to deal with telemetry generated on this host by the sources not known to a New Relic Agent.



What is this ?

But we do have an OHI and Flex ?

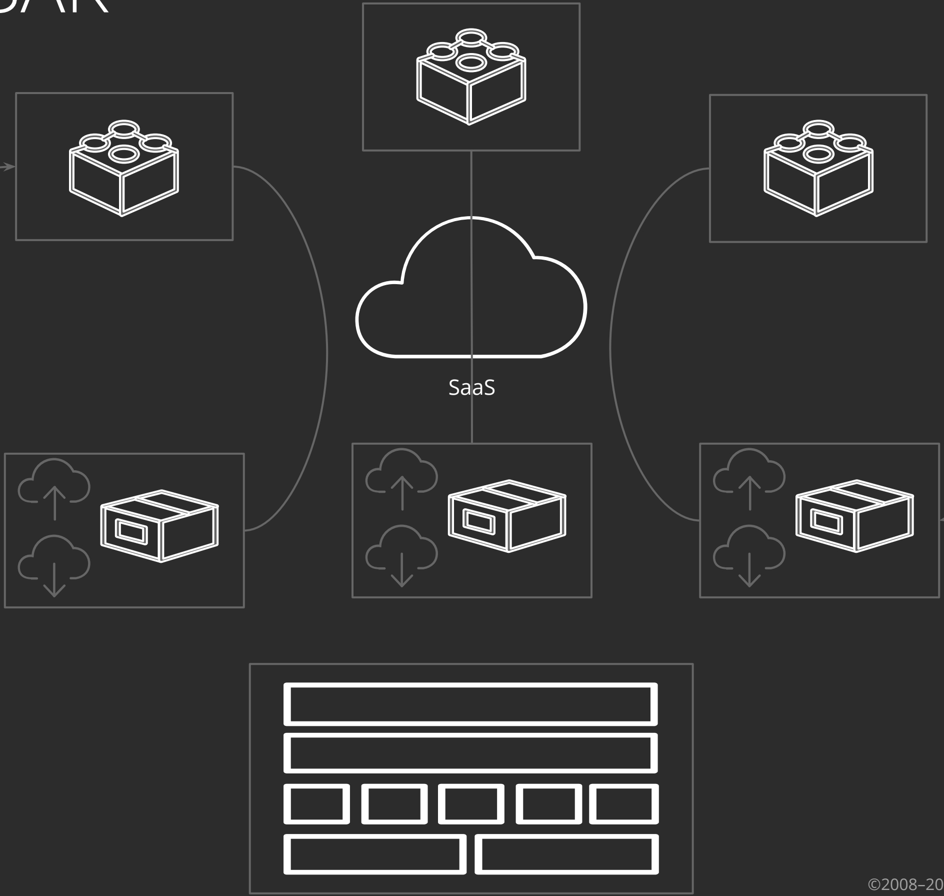
TSAK is:

❑ Not designed to be an exclusive host-bound tool. But it can be the one.

❑ Not designed to be a converter and provider of single telemetry item. But you can make it to do just that.

❑ Is designed to be a tool, which have a programmatic way to obtain, convert and deliver some data. Any data. In any supported format supported by TSAK script.

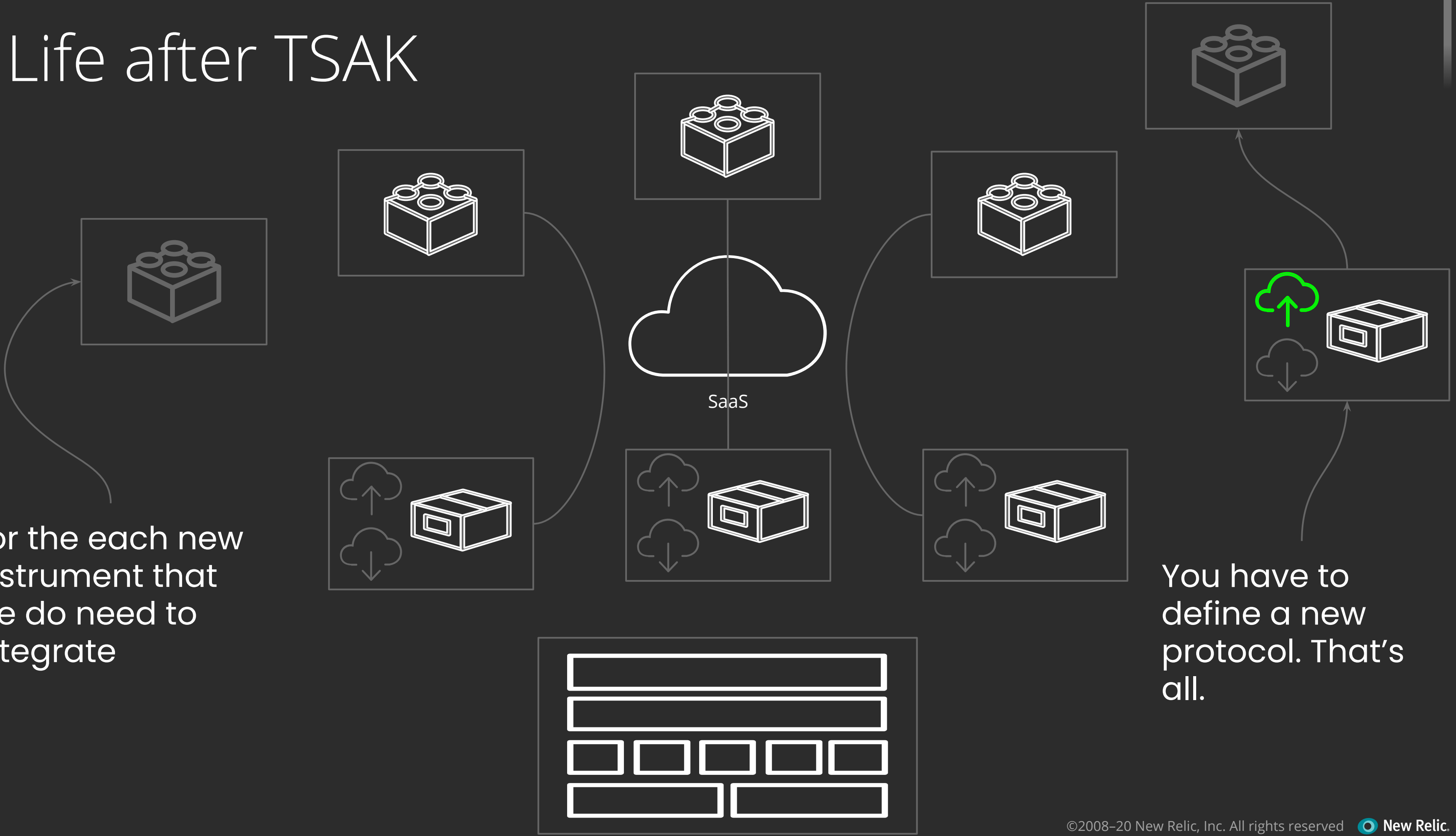
Life after TSAK



For the each integration we have

We have a tool with programmatically defined protocol, converter and feeder instruments.

Life after TSAK

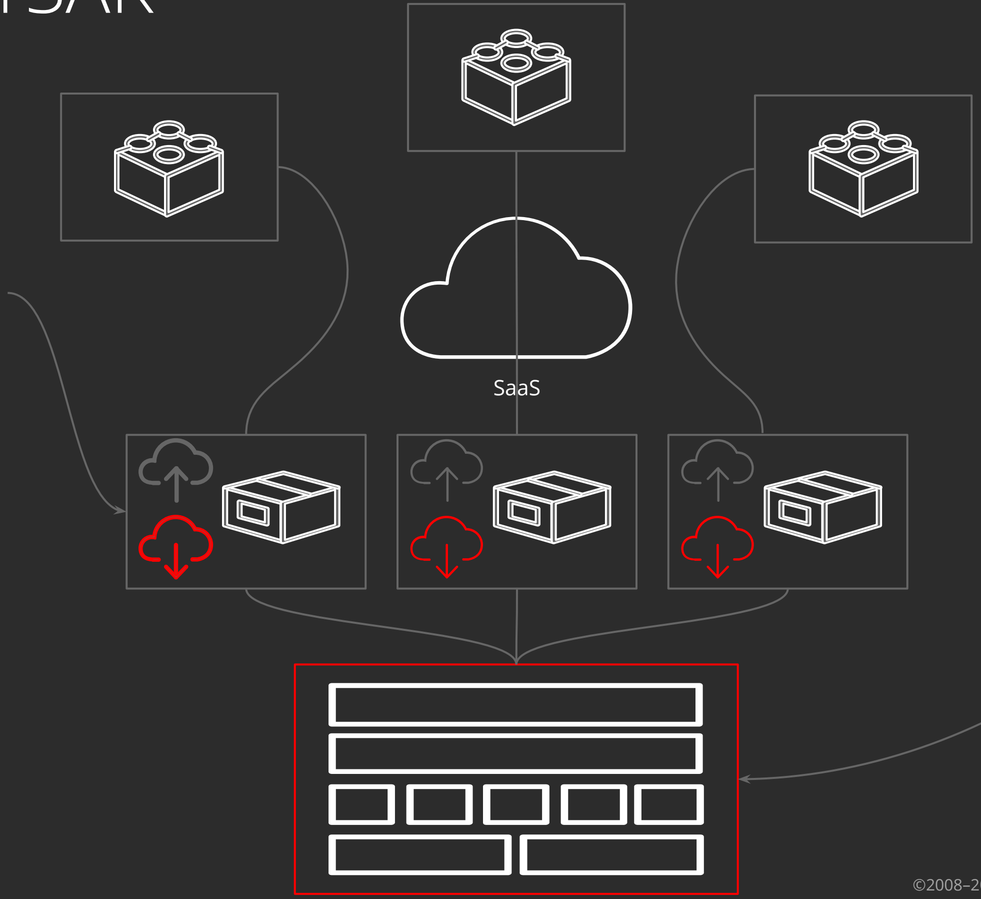


For the each new instrument that we do need to integrate

You have to define a new protocol. That’s all.

Life after TSAK

only feeder instrumentation will be affected, which dramatically cut the time for the adoption of the change

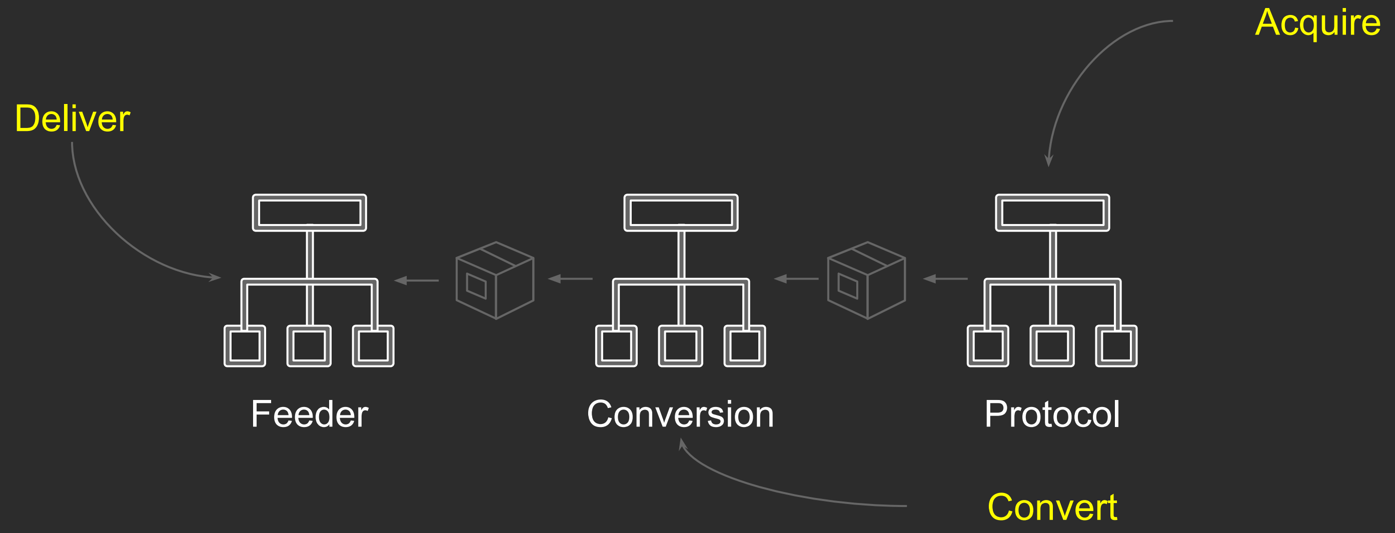


And in case of any changes on platform side

Overview of TSAK architecture

TSAK is a set of programmatic tools, written in Golang, and for the purposes of programmatic implementation of the protocol, process and feeder components, TSAK is using DSL (Domain Specific Language) called ANKO. ANKO later on will be referred as a TSAK script

Overview of TSAK architecture

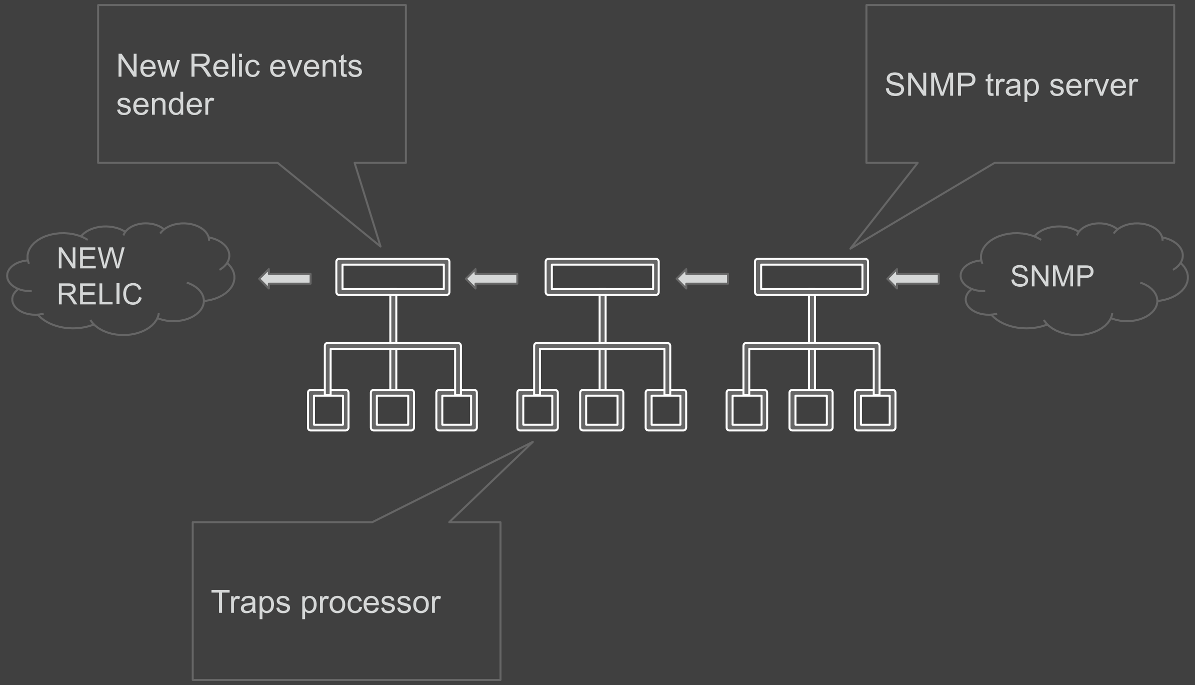


So, essentially, as a converter, TSAK have a programmatic ways for acquire, convert and deliver data where all this functionality are implemented in Go-like TSAK script.

SNMP trapper

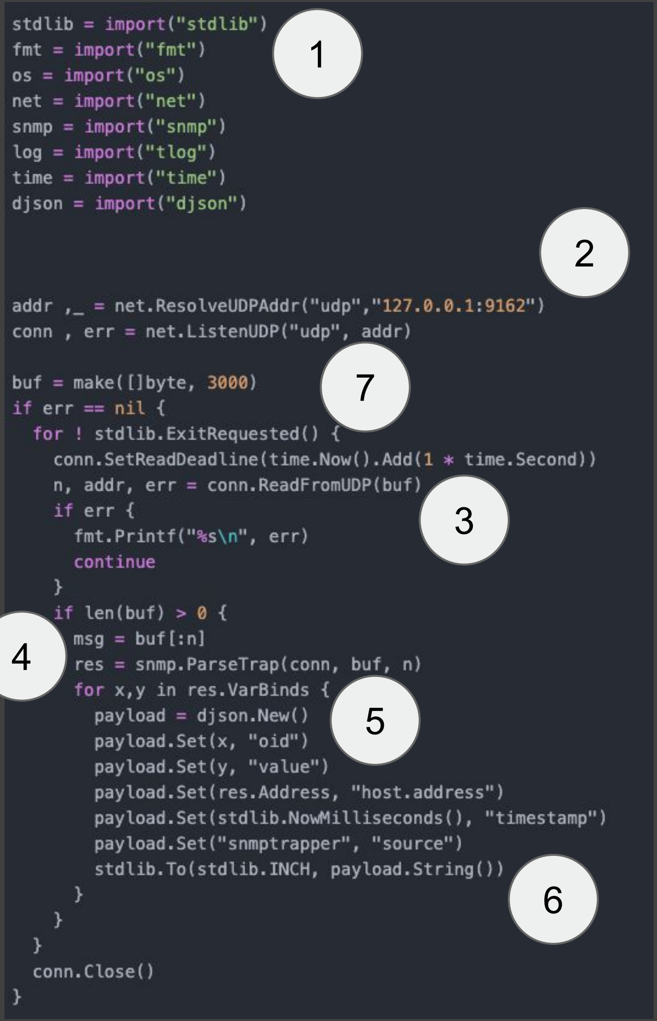
As an example, let’s review of how you bring something like an SNMP trapper/catcher using TSAK. As previously explained, TSAK creates instrument, made of three distinct parts:

1. Protocol - SNMP server;
2. Processor - all traps already pre-converted, we do need to get process complete;
3. NR sender.



SNMP trapper

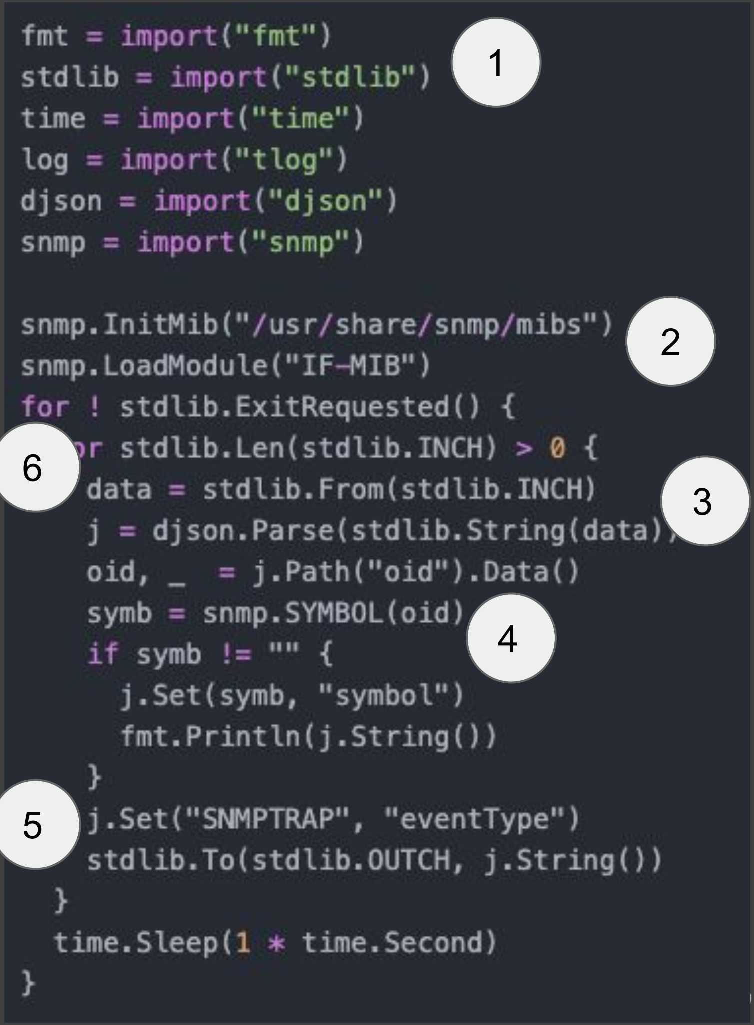
1. TSAK script imports of required modules
2. Listening on UDP socket
3. Reading from UDP
4. Parsing what we’ve got
5. Scanning SNMP VarBind
6. Sending data for later processing
7. Do all this ‘till TSAK terminates



```
stdlib = import("stdlib")
fmt = import("fmt")
os = import("os")
net = import("net")
snmp = import("snmp")
log = import("tlog")
time = import("time")
djson = import("djson")



addr ,_ = net.ResolveUDPAddr("udp","127.0.0.1:9162")
conn , err = net.ListenUDP("udp", addr)

buf = make([]byte, 3000)
if err == nil {
  for ! stdlib.ExitRequested() {
    conn.SetReadDeadline(time.Now().Add(1 * time.Second))
    n, addr, err = conn.ReadFromUDP(buf)
    if err {
      fmt.Printf("%s\n", err)
      continue
    }
    if len(buf) > 0 {
      msg = buf[:n]
      res = snmp.ParseTrap(conn, buf, n)
      for x,y in res.VarBinds {
        payload = djson.New()
        payload.Set(x, "oid")
        payload.Set(y, "value")
        payload.Set(res.Address, "host.address")
        payload.Set(stdlib.NowMilliseconds(), "timestamp")
        payload.Set("snmptrapper", "source")
        stdlib.To(stdlib.INCH, payload.String())
      }
    }
  }
  conn.Close()
}
```

SNMP trapper

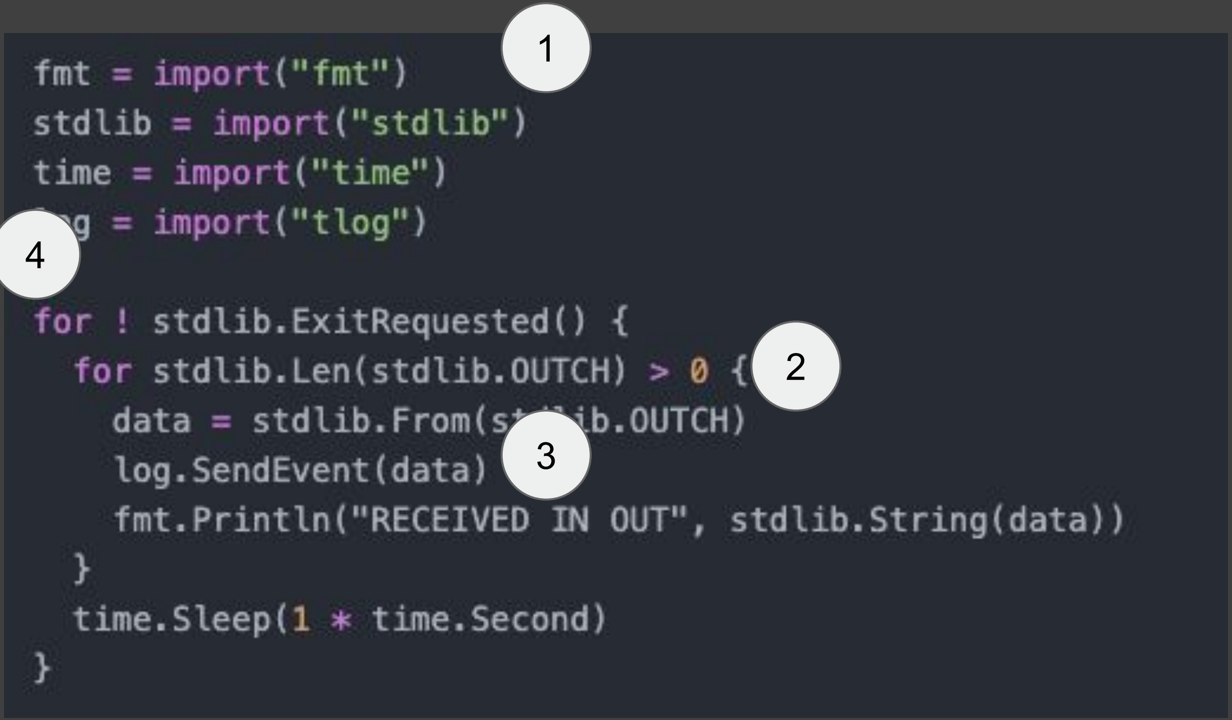
1. TSAK script imports of required modules
2. Loading the MIB’s. Because processor is a separate greenlet, it will not interfere with other components.
3. Reading from Protocol
4. Resolving SNMP Symbol
5. Push to Feeder
6. Do all this ‘till TSAK terminates



```
fmt = import("fmt")
stdlib = import("stdlib")
time = import("time")
log = import("tlog")
djson = import("djson")
snmp = import("snmp")

snmp.InitMib("/usr/share/snmp/mibs")
snmp.LoadModule("IF-MIB")
for ! stdlib.ExitRequested() {
  for stdlib.Len(stdlib.INCH) > 0 {
    data = stdlib.From(stdlib.INCH)
    j = djson.Parse(stdlib.String(data))
    oid, _  = j.Path("oid").Data()
    symb = snmp.SYMBOL(oid)
    if symb != "" {
      j.Set(symb, "symbol")
      fmt.Println(j.String())
    }
    j.Set("SNMPTRAP", "eventType")
    stdlib.To(stdlib.OUTCH, j.String())
  }
  time.Sleep(1 * time.Second)
}
```

SNMP trapper

1. TSAK script imports of required modules
2. Read from processor
3. Send event to New Relic
4. Do all this ‘till TSAK terminates



```
fmt = import("fmt")
stdlib = import("stdlib")
time = import("time")
log = import("tlog")

for ! stdlib.ExitRequested() {
  for stdlib.Len(stdlib.OUTCH) > 0 {
    data = stdlib.From(stdlib.OUTCH)
    log.SendEvent(data)
    fmt.Println("RECEIVED IN OUT", stdlib.String(data))
  }
  time.Sleep(1 * time.Second)
}
```

SNMP trapper

Then, run the tool. Here some important command-line parameters:

1. -in - location of the TSAK script for protocol side
2. -proc - location of the TSAK script for processing
3. -out - location of the TSAK script for feeder side
4. -production - run TSAK in “production mode”
5. -nrapi/-account - information necessary to communicate to NR

```
                                          4
tsak -debug -nrapi <NRAPI key> -account <NR account> -name
"trapd" -production -stdout -conf ./config.example/tsak.conf -in      1
./examples/trapd/in.script -proc ./examples/trapd/proc.script -out
      4
./examples/trapd/out.script               2                         3
```

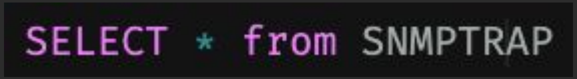
SNMP trapper

And send the trap

```
snmptrap -v 1 -c public 127.0.0.1:9162 1.2.3.4 1.2.3.4 3 0 " IF-MIB::ifOperStatus.1 i 0
```

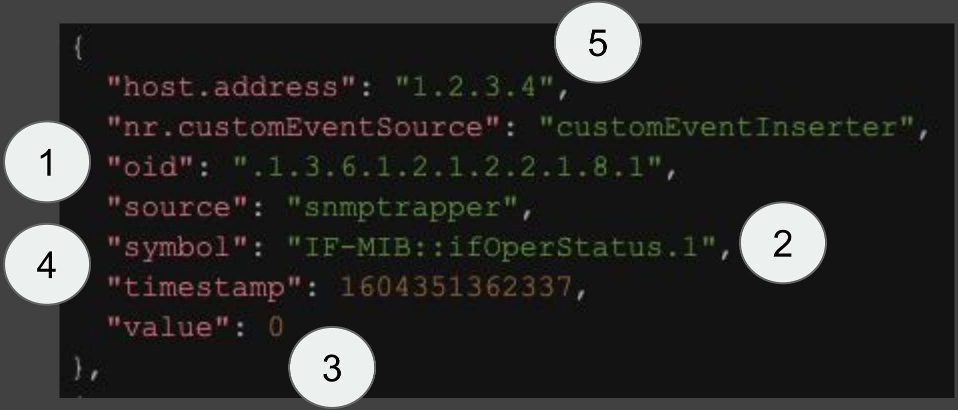
SNMP trapper

The outcome could be queried



```
SELECT * from SNMPTRAP
```

1. OID
2. Symbol
3. Value
4. Source
5. Host address



```
{
  "host.address": "1.2.3.4",
  "nr.customEventSource": "customEventInserter",
  "oid": ".1.3.6.1.2.1.2.2.1.8.1",
  "source": "snmptrapper",
  "symbol": "IF-MIB::ifOperStatus.1",
  "timestamp": 1604351362337,
  "value": 0
},
```

Thank You

vulogov@newrelic.com • @vulogov on Slack

New Relic.