SWIMming IN TCL FOR NETWORK REAL-TIME MONITORING AND MANAGEMENT OF SPACE LAUNCH AND RE-ENTRY MISSIONS

Frank Morlang

OPENACS AND TCL/TK CONFERENCE 2025, Bologna

Overview

- Introduction
- Challenge
- Realization

Introduction

- SWIM (System Wide Information Management)
  - → Air Traffic Management (ATM) Intranet
- Network Manager (NM) B2B Services
  - EUROCONTROL Network Manager (NM) system-to-system access interface
  - SWIM compliant

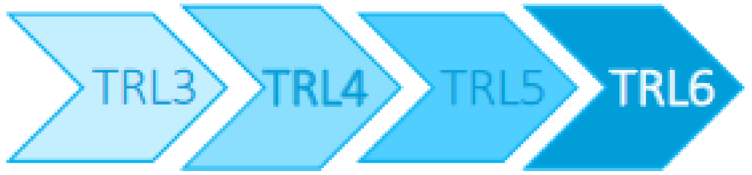
- Open ATM digital collaborative environment

Introduction

DLR

ECHO 2

- ECHO 2 (European Concept for Higher Altitude Operations Phase 2) Project
- → Three-year SESAR (Single European Sky ATM Research) 3 Joint Undertaking Project



sesar JOINT UNDERTAKING

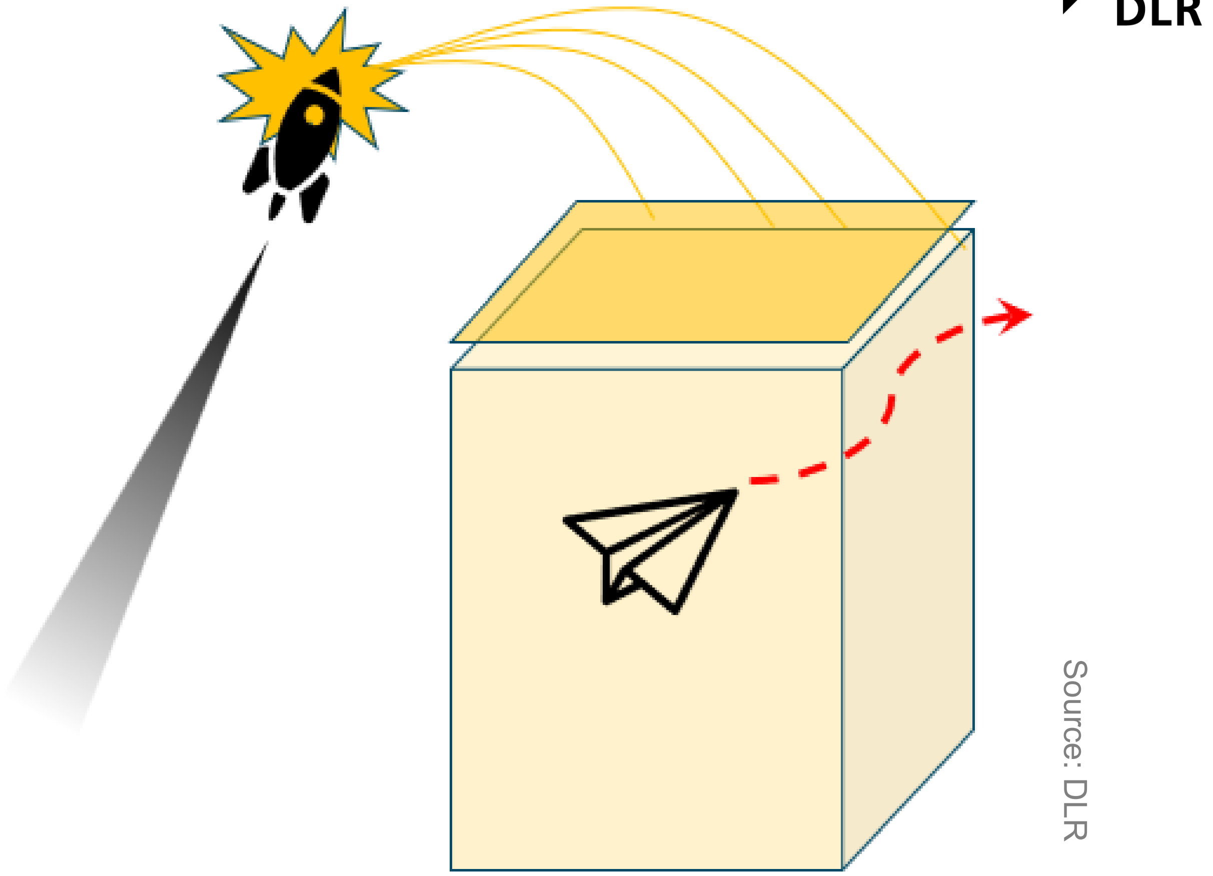
Co-funded by the European Union

Source: EUROCONTROL

- Monitoring of space launches and re-entries in real-time
- Improved situational awareness
- Air traffic flow management support

Introduction

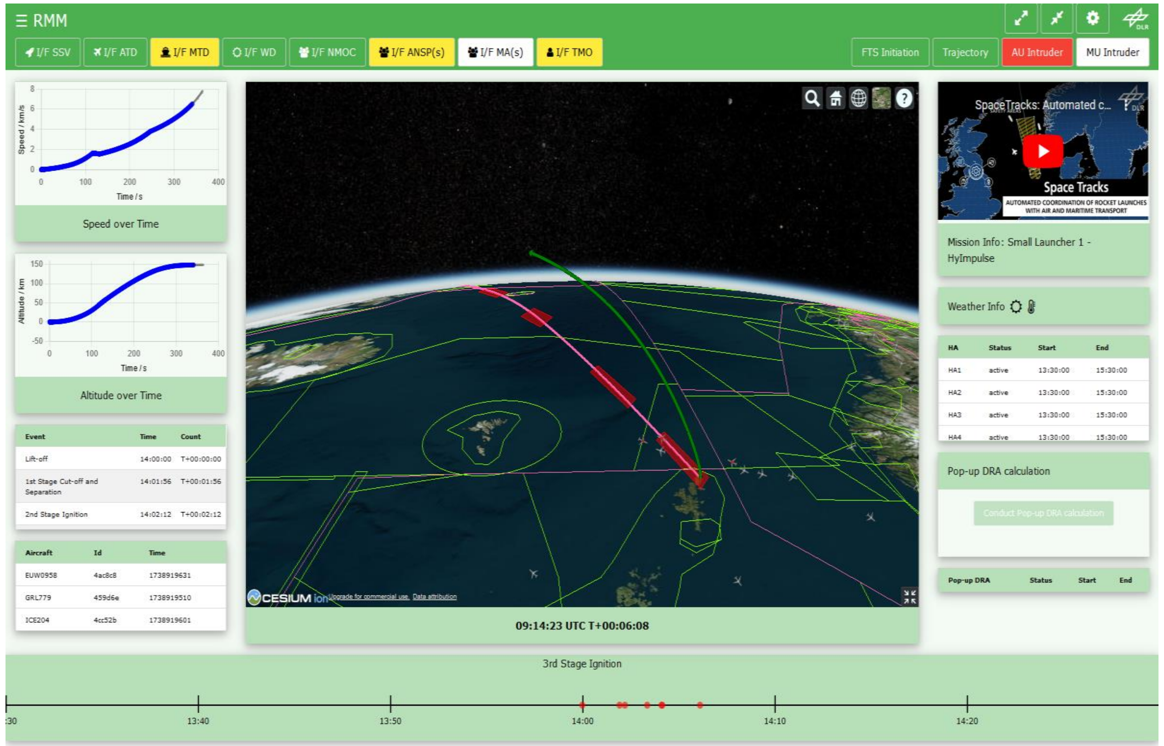
- Future space flight increase in the highly frequented and complex European airspace
- Large-scale impact on air traffic



- In case of non-nominal event:
  Activation of pre-calculated Debris Response Areas

Introduction

- N-RMM (Network Real-time Mission Monitoring) Prototype
  - Visualization of accumulated data
  - Interface to relevant SWIM NM B2B services



Event	Time	Count
Lift-off	14:00:00	T+00:00:00
1st Stage Cut-off and Separation	14:01:56	T+00:01:56
2nd Stage Ignition	14:02:12	T+00:02:12

Aircraft	Id	Time
EUW0958	4ac8c8	1738919631
GRL779	459d6e	1738919510
ICE204	4cc52b	1738919601

Mission Info: Small Launcher 1 - HyImpulse

Weather Info

HA	Status	Start	End
HA1	active	13:30:00	15:30:00
HA2	active	13:30:00	15:30:00
HA3	active	13:30:00	15:30:00
HA4	active	13:30:00	15:30:00

Pop-up DRA calculation

Conduct Pop-up DRA calculation

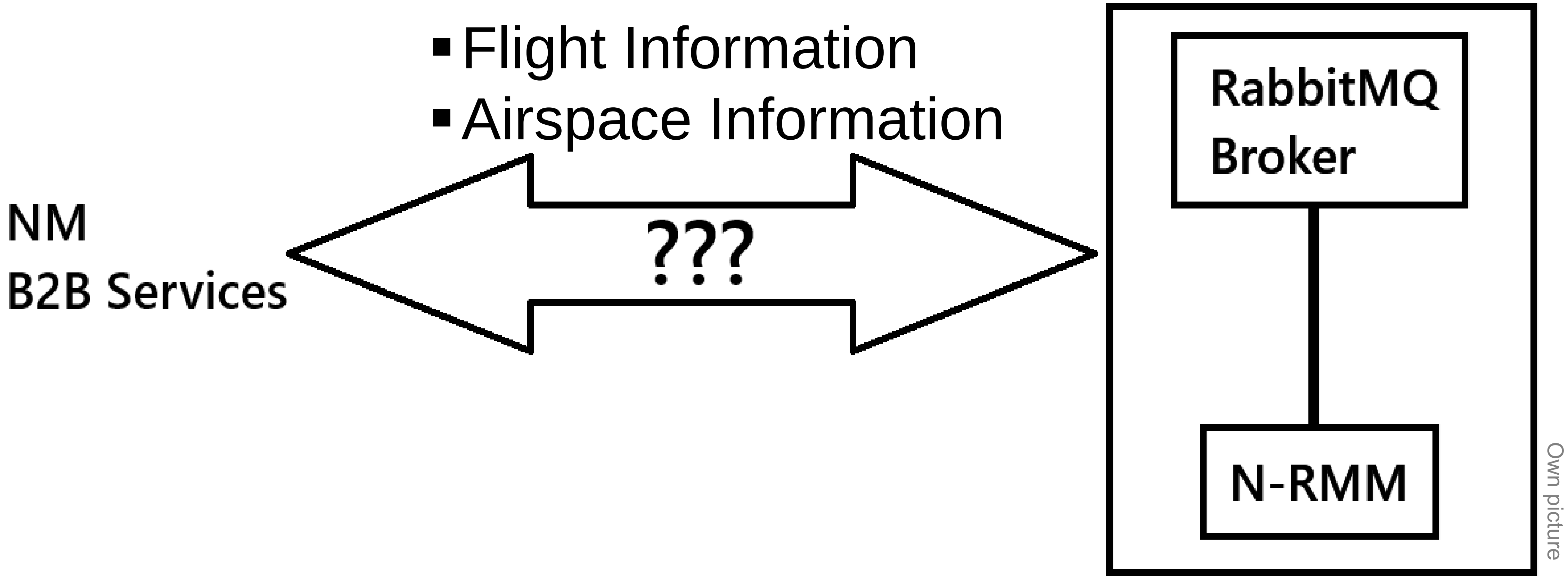
Pop-up DRA	Status	Start	End

Source: Jens Hampe, DLR

- Tailored for the needs of the European ATM Network Manager
- Tested in the context of the EUROCONTROL NM

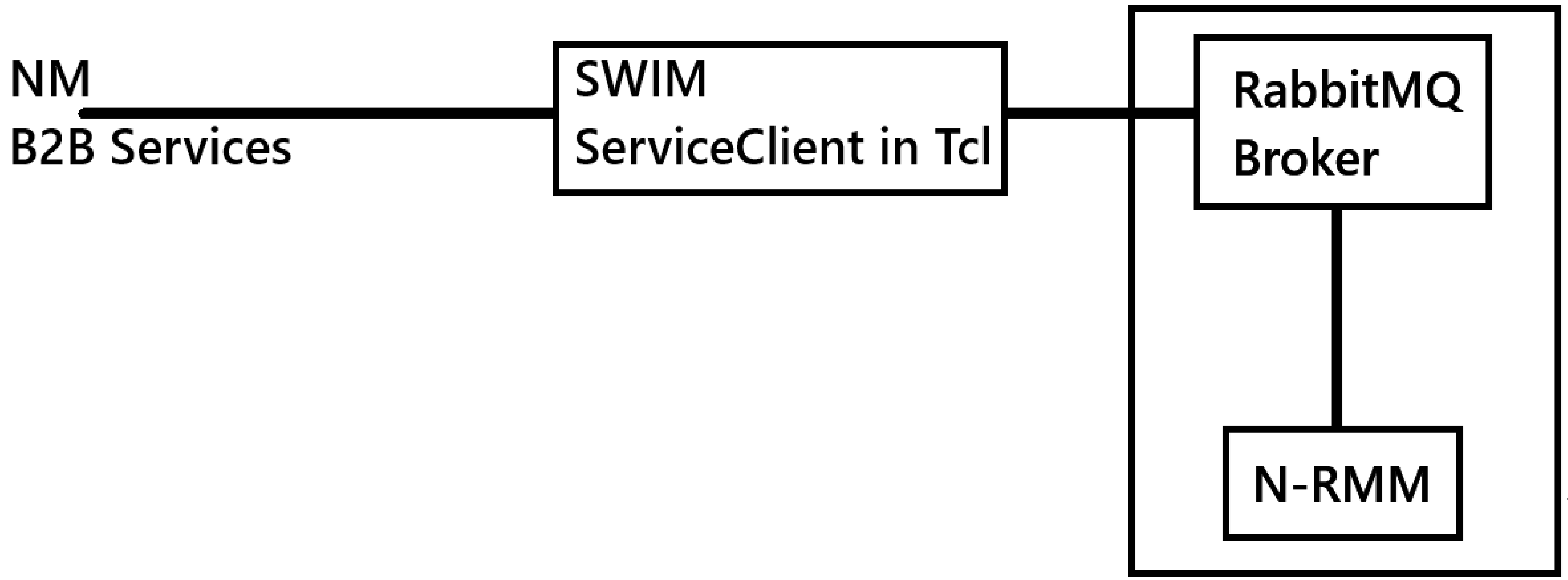
Challenge

- General Information
- Flow Information
- Flight Information
- Airspace Information



- Make an educated guess… ☺

Realization

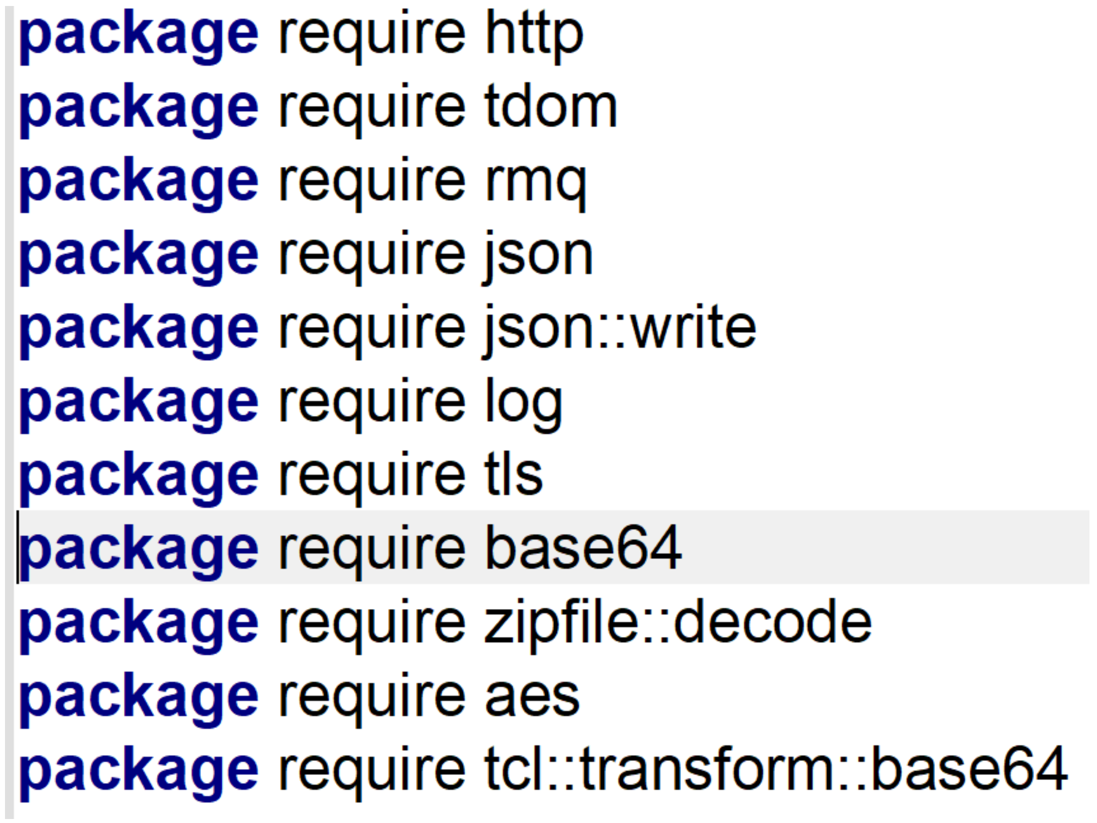


Own picture

- Pragmatic use of Tcl ☺

Realization

- Used libs:

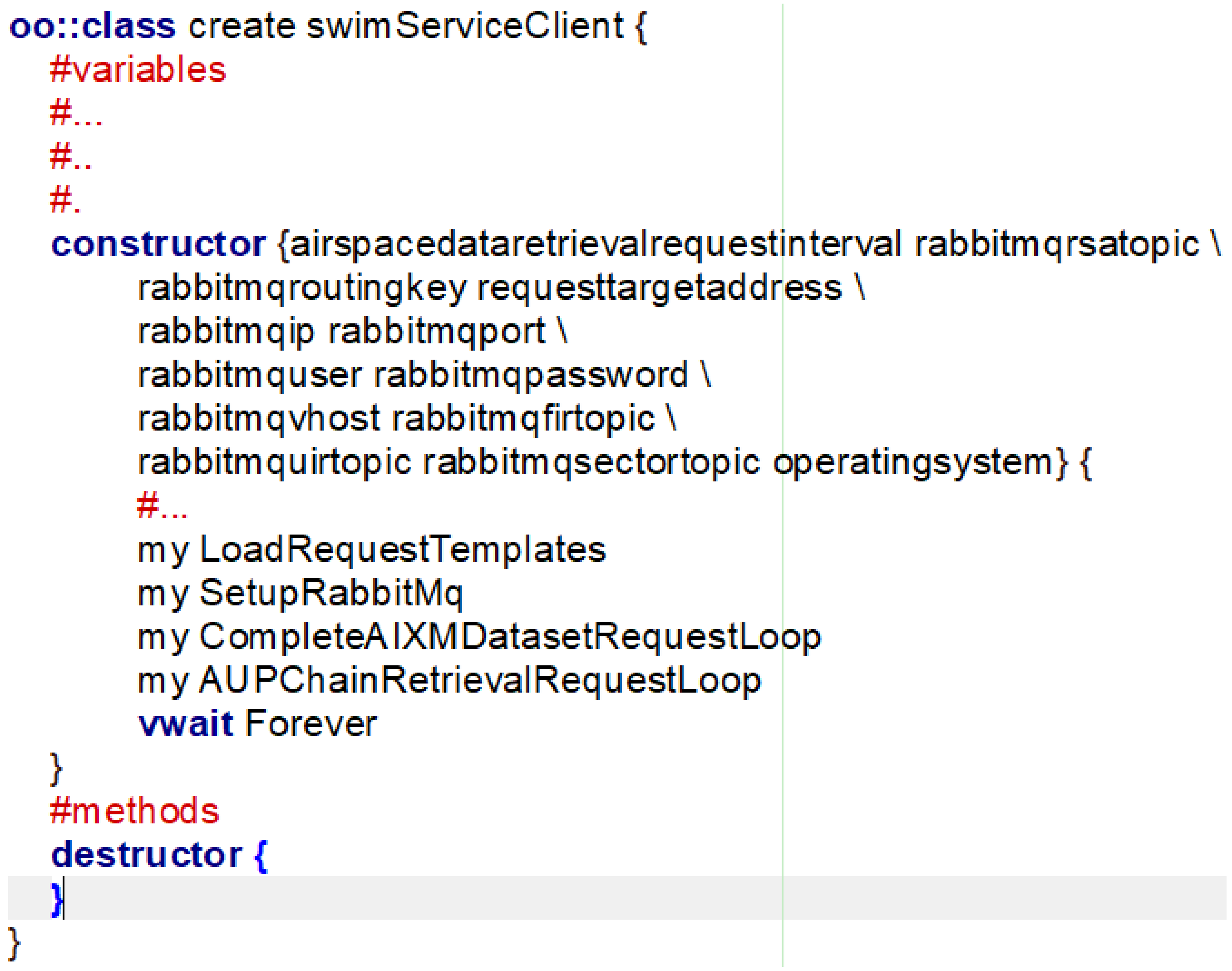


```
package require http
package require tdom
package require rmq
package require json
package require json::write
package require log
package require tls
package require base64
package require zipfile::decode
package require aes
package require tcl::transform::base64
```

- So far so good, so nice ☺

Realization

- Structure:

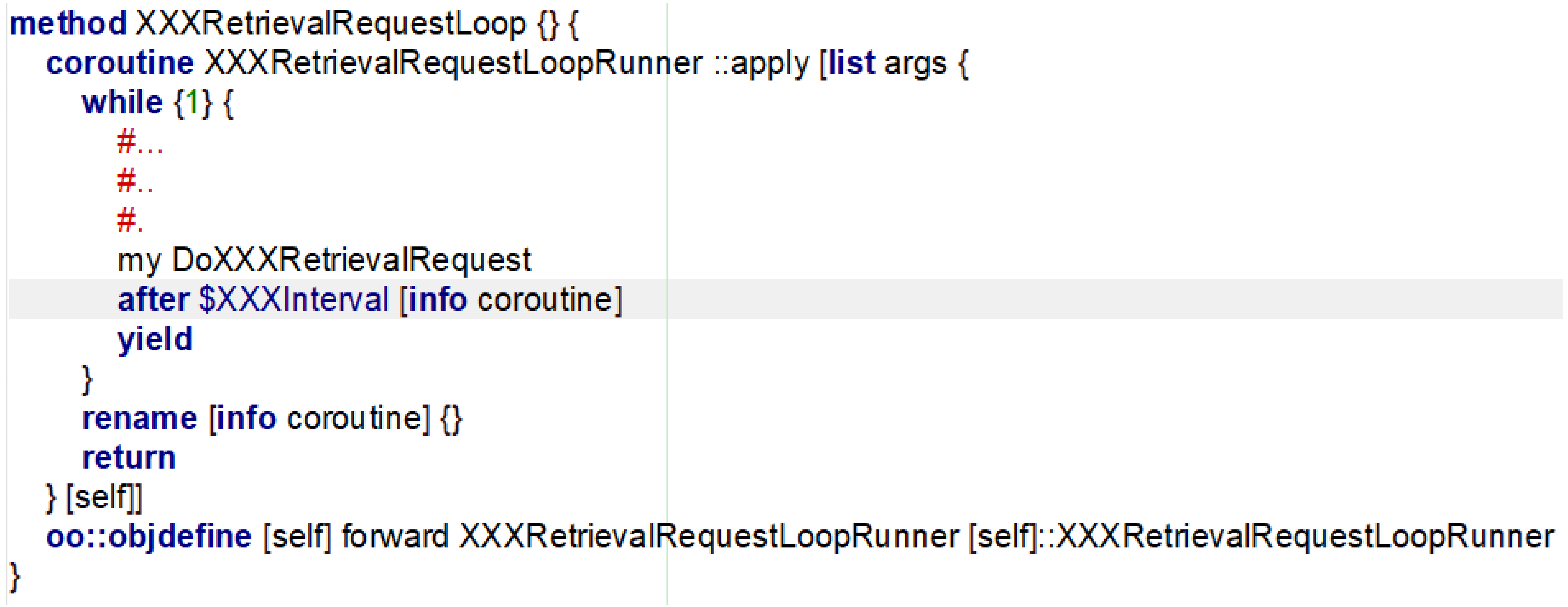


```
oo::class create swimServiceClient {
    #variables
    #...
    #..
    #.
    constructor {airspacedataretrievalrequestinterval rabbitmqrsatopic \
            rabbitmqroutingkey requesttargetaddress \
            rabbitmqip rabbitmqport \
            rabbitmquser rabbitmqpassword \
            rabbitmqvhost rabbitmqfirtopic \
            rabbitmquirtopic rabbitmqsectortopic operatingsystem} {
            #...
            my LoadRequestTemplates
            my SetupRabbitMq
            my CompleteAIXMDatasetRequestLoop
            my AUPChainRetrievalRequestLoop
            vwait Forever
    }
    #methods
    destructor {
    }
}
```

- So far so good, so nice ☺

Realization

- The RequestLoops with Coroutines:

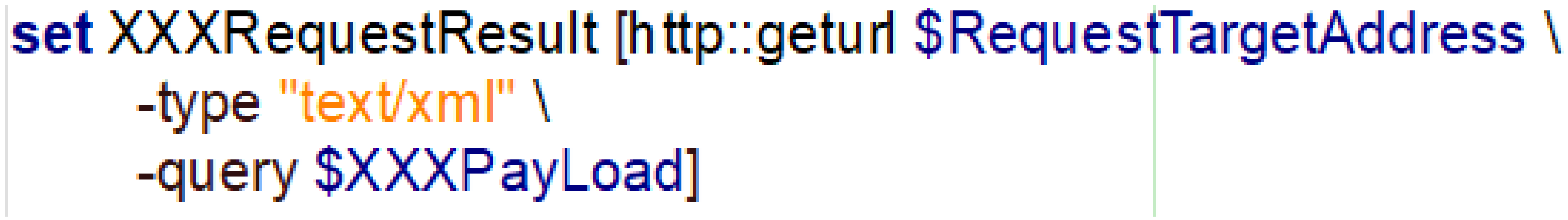


```
method XXXRetrievalRequestLoop {} {
   coroutine XXXRetrievalRequestLoopRunner ::apply [list args {
      while {1} {
         #...
         #..
         #.
         my DoXXXRetrievalRequest
         after $XXXInterval [info coroutine]
         yield
      }
      rename [info coroutine] {}
      return
   } [self]]
   oo::objdefine [self] forward XXXRetrievalRequestLoopRunner [self]::XXXRetrievalRequestLoopRunner
}
```

- So far so good, so nice ☺

Realization

- The Requests with http package:



```
set XXXRequestResult [http::geturl $RequestTargetAddress \
        -type "text/xml" \
        -query $XXXPayLoad]
```

- So far so good, so nice ☺

Realization

- Everything ☺ worked perfect!

- BUT ☹

Realization

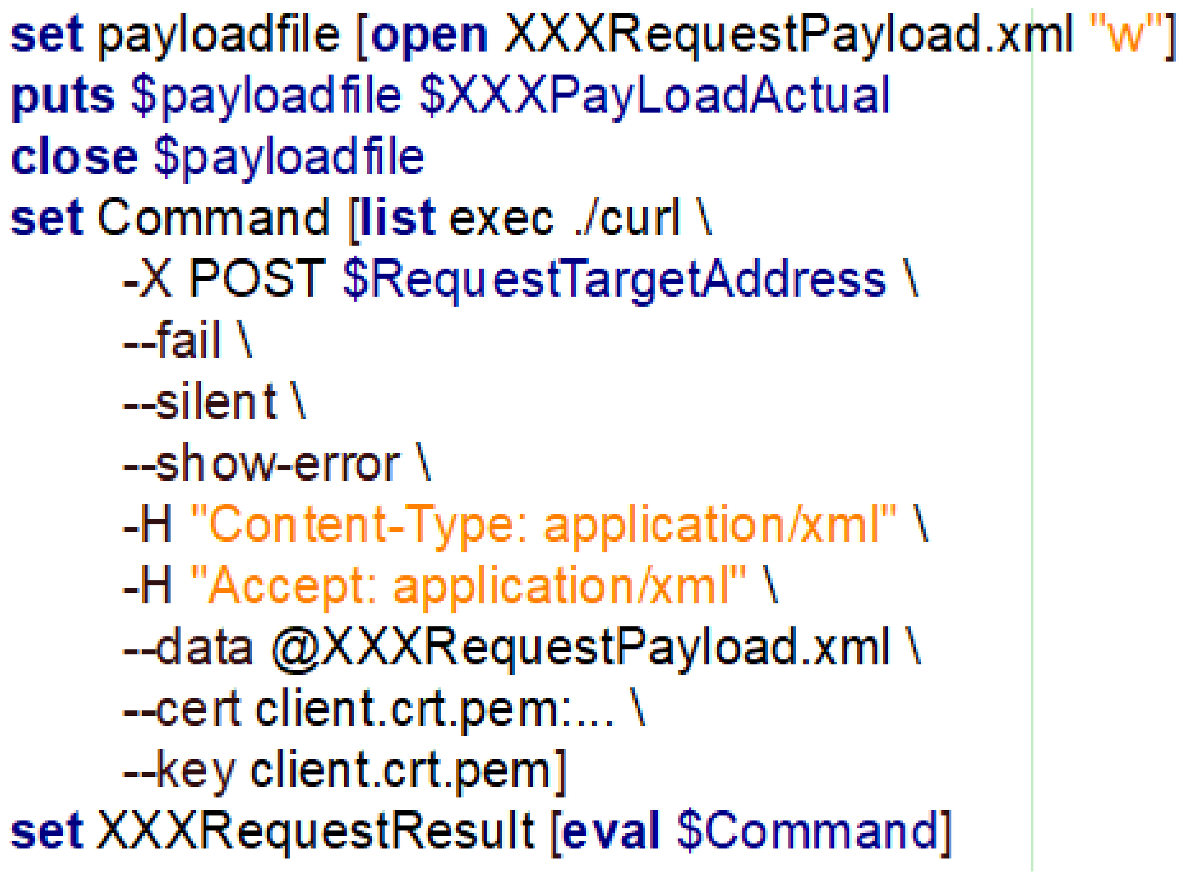
- On Windows Server installation at Eurocontrol:

error reading "sock0000022C66EFE990": software caused connection abort

- ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹ ☹

Realization

- Solution →
  exec and eval with Curl instead of using http with tls:

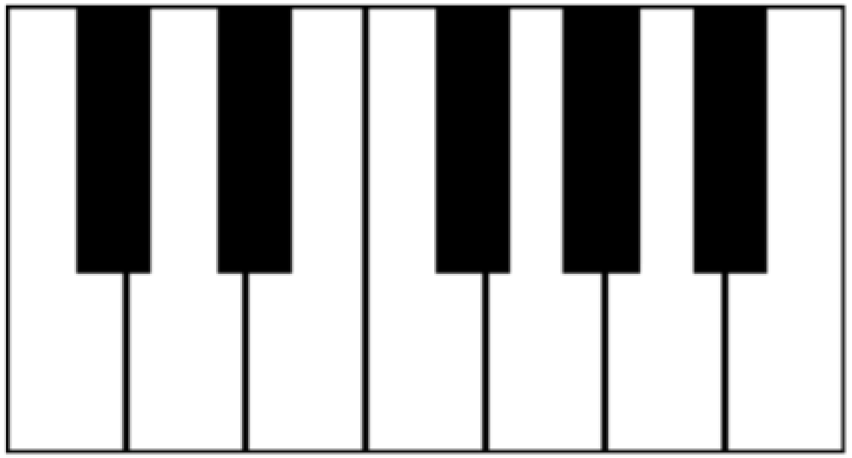


```
set payloadfile [open XXXRequestPayload.xml "w"]
puts $payloadfile $XXXPayLoadActual
close $payloadfile
set Command [list exec ./curl \
        -X POST $RequestTargetAddress \
        --fail \
        --silent \
        --show-error \
        -H "Content-Type: application/xml" \
        -H "Accept: application/xml" \
        --data @XXXRequestPayload.xml \
        --cert client.crt.pem:... \
        --key client.crt.pem]
set XXXRequestResult [eval $Command]
```

- ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺

Realization

Tcl/Tk

Project time pressure +  = SUCCESS !

- ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺

Acknowledgment

I would like to express my sincere gratitude to my colleague ***Jens Hampe****, for his scientific excellence, N-RMM work and SOAPUI service testing. His insightful feedback and encouragement was instrumental in realizing this work.*

- ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺