# Project Proposals: Infrastructure Utilities

## Contents



## Name

Infrastrucure Utilities Project

## Repo Name

infrautils

## Description

This project will offer various utilities and infrastructures for other projects to use.

The first two main utilities that the project will focus on are:

1) Counters Infrastructure

Create, update and output counters is a basic tool for debugging and generating statistics in any system. We've developed a counter infrastructure integrated into ODL which has already been successfully used with multiple products, and more recently in debugging and fixing the OpenFlow plugin/Java and LACP modules.

The following picture is an example of a log file of a system with counters integrated in it.

blocked URL

The following picture is an example of an overview of counter values using a simple REST API
blocked URL

The following picture is an example of taking the output of the counters, injecting it into an RRD, and easily earning time-series data analytics of a running application
blocked URL

2) Seamless Async Calls Utility
The decision to split a service into one or more threads with asynchronous interactions between them is frequently dependent on constraints learned late in the development and even the deployment cycle. In order to allow flexibility in making these decisions we've developed an infrastructure which is configuration driven, allowing agnostic code to be written under generic constrains which can then later be customized according to the required constraints. It also saves the developer the need to actually write the boiler-plate code to allow an async call, including: Executors, Thread Pools, Blocking Queues, Producer/Consumer Threads, etc.

Design Overview - Foo tries to invoke Bar.doSomething()

blocked URL

Presentation Slides from the ODL 2016 Developer Design Forum: http://events.linuxfoundation.org/sites/events/files/slides/Counters%20and%20Async%20Infrastructure%20-%20ODL%20format.pdf

## Scope

1) Counters Infrastructure

Every project that will want to harvest the counters capabilities, could use this infrastructure.
The entity of Counter will be modeled using YANG, which will enable future visibility using run-time MBeans.

2) Seamless Async Calls Utility

blocked URL

"Fat" applications that wish to allow dynamic and seamless **in-app** async communication could use this infrastructure.
Future version of MD-SAL, where RPCS will take a Callback instead of a Future return value, could use the same infrastructure by Proxying the Callback.

## Resources Committed (developers committed to working)

Guy Sela (guy.sela@hpe.com) [gerrit id: guys]

Ravit Perez (ravit.peretz@hpe.com)

Tomer Pearl (tomer.pearl@hpe.com)

## Initial Committers

Guy Sela (guy.sela@hpe.com) [gerrit id: guys]

Ravit Perez (ravit.peretz@hpe.com)

Tomer Pearl (tomer.pearl@hpe.com)

Robert Varga (rovarga@cisco.com)

## Vendor Neutral

No vendor package names in code
No vendor branding present in code or output of build
No vendor branding present in documentation

## Meets Board Policy (including IPR)

TBD