



## Technical Manual of the US–Common Layer

**Organisation:** Copyright (C) 2019-2020 Olivier Boudeville

**Contact:** about (dash) us-common (at) esperide (dot) com

**Creation date:** Saturday, May 2, 2020

**Lastly updated:** Friday, May 15, 2020

**Status:** Work in progress

**Version:** 0.0.1

**Dedication:** Users and maintainers of the US–Common layer.

**Abstract:** The role of the [US-Common](#) layer (part of the [Universal Server](#) project) is to provide base elements on which the various *Universal Services* are built, notably:

- the Universal Server itself: see [us-main](#)
- the Universal Webserver: see [us-web](#)

We present here a short overview of these services, to introduce them to newcomers.

The next level of information is to read the corresponding [source files](#), which are intensely commented and generally straightforward.

## Table of Contents

<b>Technical Manual of the US-Common Layer</b>	<b>1</b>
Overview	3
Layer Stack	3
Facilities Provided by this Layer	4
Licence	4
Current Stable Version & Download	4
Using Cutting-Edge GIT	5
Using OTP-Related Build/Runtime Conventions	5
Support	6
Please React!	6
Ending Word	6

## Overview

The [US-Common](#) layer is the basis (lowest-level) of the [Universal Server](#) project.

Its purpose is to provide base elements on which the various *Universal Services* are built, notably:

- the Universal Server itself: see [us-main](#)
- the Universal Webserver: see [us-web](#)

We present here a short overview of these services, to introduce them to newcomers.

The next level of information is to read the corresponding [source files](#), which are intensely commented and generally straightforward.

## Layer Stack

From the highest level to the lowest, as shown [here](#), usually a software stack involving US-Common is structured that way:

- applicative layer such as [us-main](#), [us-web](#), etc.
- [US-Common](#) (this layer)
- [Ceylan-Traces](#) (for advanced runtime traces)
- [Ceylan-WOOPER](#) (for OOP)
- [Ceylan-Myriad](#) (as an Erlang toolbox)
- [Erlang](#) (for the compiler and runtime)
- [GNU/Linux](#)

The shorthand for US-Common is `uc`.

## Facilities Provided by this Layer

These are mainly common services centralised here so that the various US applications can make use of them:

- **USServer**: a general **abstraction of a server**, so that all US ones inherit the corresponding base features (ex: name registration, uptime information, applicative ping, state description, etc.)
- **USConfigServer**: a server (usually a singleton) in charge of **managing all US-level configuration information** on behalf of the other US servers; this comprises the look-up, parsing and checking of the relevant configuration files, the setting of the corresponding information then made available to the rest of the US framework (EPMD port, TCP port range, cookie, execution context, application and log directories, name and scope of registrations, user/group information, etc.)
- **USScheduler**: a server whose purpose is to **schedule any kind of asynchronous, independent tasks** (think: "crontab on steroids"); it allows planning task commands to be issued to actuators one time, multiple ones, or indefinitely, based on user-level periods with various policies, on a best-effort basis yet reliably (proper time and timer management), trying to find a balance between the respect of the requested periodicities and the correction of any delay incurred (see also a [corresponding test](#) of it)
- **UStaskRing**: a facility useful to **schedule a set of periodic tasks synchronously** (no overlapping between them) **and uniformly** (as evenly as possible over time); typically useful to pace regularly a set of actions of indefinite number that are ruled by a common periodicity and/or to share a resource unable to cope with concurrent accesses (ex: a non-reentrant third-party log analysis tool that maintains its own opaque state on filesystem, yet have to operate on a set of virtual hosts)

## Licence

US-Common is licensed by its author (Olivier Boudeville) under the [GNU Affero General Public License](#) as published by the Free Software Foundation, either version 3 of this license, or (at your option) any later version.

This allows the use of the US-Common code in a wide a variety of software projects, while still maintaining copyleft on this code, ensuring improvements are shared.

We hope indeed that enhancements will be back-contributed (ex: thanks to merge requests), so that everyone will be able to benefit from them.

## Current Stable Version & Download

We prefer using GNU/Linux, sticking to the latest stable release of Erlang, and building it from sources, thanks to GNU make.

As mentioned, the single, direct prerequisite of **US-Common** is [Ceylan-Traces](#), which implies in turn [Ceylan-WOOPER](#), then [Ceylan-Myriad](#) and [Erlang](#).

Refer to the corresponding [Myriad prerequisite section](#) for more precise guidelines, knowing that US-Common does not need modules with conditional support such as `crypto` or `wx`.

## Using Cutting-Edge GIT

This is the installation method that we use and recommend; the US-Common master branch is meant to stick to the latest stable version: we try to ensure that this main line always stays functional (sorry for the pun). Evolutions are to take place in feature branches and to be merged only when ready.

Once Erlang is available, it should be just a matter of executing:

```
$ git clone https://github.com/Olivier-Boudeville/Ceylan-Myriad myriad
$ cd myriad && make all && cd ..

$ git clone https://github.com/Olivier-Boudeville/Ceylan-WOOPER wooper
$ cd wooper && make all && cd ..

$ git clone https://github.com/Olivier-Boudeville/Ceylan-Traces traces
$ cd traces && make all && cd ..

$ git clone https://github.com/Olivier-Boudeville/us-common
$ cd us-common && make all
```

Running a corresponding test just then boils down to:

```
$ cd test && make class_USScheduler_run CMD_LINE_OPT="--batch"
```

Should LogMX be installed and available in the PATH, the test may simply become:

```
$ make class_USScheduler_run
```

## Using OTP-Related Build/Runtime Conventions

As discussed in these sections of [Myriad](#), [WOOPER](#) and [Traces](#), we added the (optional) possibility of generating a US-Common *OTP application* out of the build tree, ready to be integrated into an (*OTP*) *release*. For that we rely on [rebar3](#), [relx](#) and [hex](#).

Unlike Myriad (which is an OTP *library* application), US-Common is (like WOOPER and Traces) an OTP *active* application, meaning the reliance on an application that can be started/stopped (`us_common_app`), a root supervisor (`us_common_sup`) and, here, two proper supervisor bridges (`us_common_scheduler_bridge_sup` and `us_common_config_bridge_sup`).

As for Myriad, WOOPER and Traces, most versions of US-Common are also published as [Hex packages](#).

For more details, one may have a look at [rebar.config.template](#), the general rebar configuration file used when generating the US-Common OTP application and release (implying the automatic management of Myriad, WOOPER and Traces).

Once the steps listed in [Using Cutting-Edge GIT](#) have been performed, an OTP-level testing can be done as well, based on `us_common_otp_application_test`; from the root of US-Common:

```
$ cd test && make us_common_otp_application_run CMD_LINE_OPT="--batch"
```

All relevant OTP applications shall be started in turn, US-Common shall then be ready for use, until all these applications are orderly shutdown.

## **Support**

Bugs, questions, remarks, patches, requests for enhancements, etc. are to be reported to the [project interface](#) (typically [issues](#)) or directly at the email address mentioned at the beginning of this document.

## **Please React!**

If you have information more detailed or more recent than those presented in this document, if you noticed errors, neglects or points insufficiently discussed, drop us a line! (for that, follow the [Support](#) guidelines).

## **Ending Word**

Have fun with US-Common!

