# Technical Manual of the `US-Common` Layer




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

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

**Lastly updated:** Wednesday, February 22, 2023

**Version:** 0.1.1

**Status:** Stable

**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 either to browse the US-Common API documentation or simply to read the corresponding source files, which are intensely commented and generally straightforward.

The latest version of this documentation is to be found at the official US-Common website (`http://us-common.esperide.org`).

The documentation is also mirrored here.

# Table of Contents

## 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.

The project repository is located here.

## Layer Stack

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

- an applicative layer such as US-Main or 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)

# Configuration

## Server Configuration

The configuration of the Universal Server infrastructure lies primarily in a dedicated `us.config` file, which is searched from various base directories, according to the following order:

1. in any base directory designated by the standard `XDG_CONFIG_HOME` environment variable, otherwise in default `~/.config`
2. in any of the base directories listed (separator being `:`) in the standard `XDG_CONFIG_DIRS` environment variable, otherwise in default `/etc/xdg`

Each of these base directories is searched in turn for a `universal-server` subdirectory that would contain a `us.config` file, and the first found one is chosen as the **US Configuration directory**. Any other US-related configuration file is then expected to be found in the same directory.

In practice, often the `~/.config/universal-server/us.config` location is preferred.

All US configuration files are in the ETF format (for *Erlang Term Format*).

One may refer to this example us.config to learn their structure and derive one's own `us.config`.

### Client Configuration

Each US service (ex: `US-Main`, `US-Web`, etc.) can be monitored (locally or remotely) thanks to a corresponding `priv/bin/monitor-us-*.sh` script, which must be given the necessary information (hostname, cookie, TCP port range, etc.) in order to contact the target US instance.

This information is typically stored in a `us-*-remote-access.config` ETF file, located as well in the aforementioned US configuration directory.

## 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

In general, 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`.

Most uses of US-Common will require `authbind` (ex: on Arch Linux, obtained from the AUR, typically with thanks to the AUR installer that Ceylan-Hull recommends and installs ).

### 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 ..

# Note the dash becoming an underscore, for OTP compliance:
$ git clone https://github.com/Olivier-Boudeville/us-common 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`).

## 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!


US Common