
# Writing and Using Platforms and Models in C++ User Guide

Imperas Software Limited
Imperas Buildings, North Weston,
Thame, Oxfordshire, OX9 2HA, UK
docs@imperas.com


| | |
|---|---|
| Author: | Imperas Software Limited |
| Version: | 2.0.0 |
| Filename: | Writing_and_Using_Platforms_and_Models_in_CPP_User_Guide.doc |
| Project: | Writing and Using Platforms and Models in C++ User Guide |
| Last Saved: | Monday, 13 January 2020 |

# Copyright Notice



## Right to Copy Documentation



## Destination Control Statement



## Disclaimer

Table of Contents

# 1 Preface

The Imperas simulators can use models described in C or C++ and the models can be exported to be used in simulators and platforms using C, C++, SystemC or SystemC TLM2.

This introductory document describes how to use the OVP C++ APIs to write simple platforms and to control the simulator with test harnesses also written in C++.

It describes how the OVP OP C++ API is used in C++ programs for use with Imperas and OVP virtual platform simulators.

## *1.1 Related Documentation*

This documents the OP C++ API. This is built on the OP C API. You need to ensure you are aware of that API by looking through its introductory document:

- Writing Platforms and Modules in C User Guide

You also need to be aware of its Doxygen API online documentation in your installation: This Doxygen documentation of the OP C API is available at:

```
IMPERAS_HOME/doc/api/op/html/index.html
```

There are two related documents that focus on controlling the simulators using the OVP OP C API. If your interest is running simulations of existing platforms/models, more advanced test harnesses, or debugger/3rd party simulator integration then these documents should be your focus:

- Simulation Control of Platforms and Modules User Guide
- Advanced Simulation Control of Platforms and Modules User Guide

If you are creating platforms or subsystems/modules of components, then you need to look at using iGen. There are two documents that describe the use of iGen to create platforms and modules using its advanced, high level input script. With iGen script, larges, complex platforms and modules can be created with very few lines of code:

- iGen Model Generator Introduction
- iGen Platform and Module Creation User Guide

There are several other relevant documents available:

Getting Started

- Imperas Installation and Getting Started Guide

Interface, API, and iGen related

- OVP Peripheral Modeling Guide
- OVPsim Using OVP Models in SystemC TLM2.0 Platforms
- Imperas Peripheral Generator Guide (using iGen)

## 1.2 Notation

Code Text representing code, a command or output.
*keyword* A word with special meaning.

## 1.3 Glossary / Terminology

**OP C API** - **O**VP **P**latforms API - C API used for creating and controlling virtual platforms. 2nd generation API, replaces ICM API. iGen creates modules/platforms in C using this API.

**OP C++ API** - **O**VP **P**latforms API - C++ API used for creating and controlling virtual platforms. Built using OP C API.

**iGen** - Imperas productivity tool that has a powerful script based function API that is used to create C/C++/SystemC models and templates. Described in the iGen Model Generator Introduction, and for platforms/modules, in the iGen Platform and Module Generator User Guide.

**OVPsim** - Simulator for Open Virtual Platforms that executes platforms and models coded in the OVP APIs

**CpuManager** - Imperas commercial simulator that fully implements the APIs defined by OVP (OVPsim implements a subset)

**Platform / Module** – a collection of components connected together into a level of hierarchy in a system to be simulated. This is a program in C/C++ making calls into OP API and normally compiled into a shared object/dynamically linked library and loaded by the simulator at run time.

**Testbench / Harness** – (used interchangeably) – A program in C/C++ making calls into the OP API to connect and control OVP components. It is normally linked to the simulator to provide an .exe binary that can be executed. Used to instance one or more platforms/modules and controls their execution. The main difference, from a platform/module, is that a testbench or harness includes a definition of the function main(), may include a command line parser and is linked to create an executable binary (.exe) file.

**Root Module** - used to describe the initial platform/module that instances one or more platforms/modules and controls their execution. Used in the testbench / harness.

# 2 Introduction

Imperas simulation technology enables very high performance simulation, debug and analysis of platforms containing multiple processors and behavioral peripheral models. The technology is designed to be extensible: you can create your own platforms, new models of processors, and other platform components using interfaces and libraries supplied by Imperas. Platform models developed using this technology can be used both with Imperas simulation products and the freely-available OVPsim platform simulator.

## 2.1 Prerequisites

Since harnesses and test benches for use with Imperas and OVP tools are written in C, an important prerequisite is that you must be proficient in the C language. If you want to use C++ then it is expected that you are proficient in the use of C++ and how it uses a C API.

## 2.2 Obtaining & Installing the OP C++ API

The OP C++ API is part of all Imperas / OVP installations and thus you should already have it installed and be ready for use.

## 2.3 Compiling Examples described in this Document

The examples use modules, processors, component models and tool chains, available to download from the www.OVPworld.org website or as part of an Imperas installation.

The compilation of the examples utilize a Makefile, the instructions for which indicate the use of the command *make*, on Windows systems the MinGW *mingw32-make* command should be used in its place.

The Makefiles referred to in this document are written for GNU make. Standard Makefiles supplied by Imperas support compilation and linking using GNU tools on both Windows and Linux.

Example scripts will be referred to as (for example) *example.sh*, this being the extension used on Linux or for Windows MSYS shells. On Windows the script would be called *example.bat*

## 2.4 Shared Objects and Executables

The shared objects referred to in this document are either Linux shared objects, with suffix *.so* or Windows dynamic link libraries with suffix *.dll*.

The executables referred to in this document are either Linux or Windows programs and have the suffix *.exe*

# 3 Document Status (May 2016)

This document is currently under construction.

Its purpose is to show users how to use the C++ version of the OP API to build and control their platforms.

The document will introduce the different header files, and then walk through the concepts used, and then show in detail several examples that demonstrate specific features.

## *3.1 The headers related to C++ are:*

ImpPublic\include\host\op\op.hpp

### 3.1.1 Processor specific headers (examples) are:

ImperasLib\source\ovpworld.org\processor\or1k\1.0\cpp\generic.hpp
ImperasLib\source\arm.ovpworld.org\processor\arm\1.0\cpp\ arm_Cortex-A57MPx1.hpp
ImperasLib\source\arm.ovpworld.org\processor\arm\1.0\cpp\ arm_Cortex-A57MPx4.hpp

## *3.2 Examples currently in the release are:*

- Examples\SimulationControl\armStopReasonInCPP
- Examples\SimulationControl\behaviorAndControlInCPP
- Examples\SimulationControl\minimalHarnessUsingCPP
- Examples\SimulationControl\dmiInCPP

## *3.3 Still needed in a demo package is:*

- OP C++ ARMv8 FM booting Linux

#