# Acronyms and Definitions

Version 1.5

## ***Revision History***

| Revision | Description of Change | Date |
|---|---|---|
| v1.0 | Initial creation for OpenCPI 1.0 | 2/2016 |
| v1.1 | Reorganized and updated for OpenCPI 1.1 | 3/2017 |
| v1.2 | Updated for OpenCPI Release 1.2 | 8/2017 |
| v1.3 | Updated for OpenCPI Release 1.3 | 2/2018 |
| v1.4 | Updated for OpenCPI Release 1.4 | 9/2018 |
| v1.5 | Updated for OpenCPI Release 1.5 | 4/2019 |

## ***Document Conventions***

This document uses *italic type* to indicate a keyword that is defined elsewhere, and possibly later, within.

## 1 Acronyms

**ACI** *Application* Control Interface

**ARM** Advanced RISC Machine

**AV** ANGRYVIPER Team: sometimes used as prefix on ticket numbers within code

**AXI** *Advanced eXtensible Interface*

**BSP** *Board Support Package*

**CDK** *Component Development Kit*

**CPU** Central Processing Unit

**DSP** Digital Signal Processing or Digital Signal Processor

**FPGA** Field Programmable Gate Array

**GPP** General Purpose Processor

**GPU** Graphics Processing Unit

**HDL** *Hardware Description Language*

**OAS** OpenCPI *Application Specification*

**OCL** OpenCL

**OCS** OpenCPI *Component Specification*

**OHAD** OpenCPI HDL *Assembly Description*

**OpenCL** Open Computing Language

**OpenCPI** Open Component Portability Infrastructure

**OPS** OpenCPI *Protocol Specification*

**OSS** Open Source Software

**OWD** OpenCPI *Worker Description*

**PCI** Peripheral Component Interconnect

**PCIe** *PCI*-Express

**RCC** Resource Constrained C-Language: see *RCC Authoring Model*

**RPM** RPM Package Manager

**UUT** Unit Under Test

**VHDL** VHSIC Hardware Description Language

**VM** Virtual Machine

**XML** eXtensible Markup Language

**ZLM** *Zero Length Message*

## 2 Definitions

**Accessibility**
See *configuration property accessibility.*

**Adapter Worker**
*Worker* used when two connected workers are not connectable in some way due to different interface choices in the *OWD*. Adapter workers are normally inserted automatically as needed, *e.g.* between a *worker* that has a 16-bit bus and one with a 32-bit one.

**Advanced eXtensible Interface (AXI)**
Industry-standard bus used by ARM processors.

**Application**
[noun] In this context of Component-Based Development, an *application* is a composition or assembly of *components* that, as a whole, perform some useful function.

[adjective] The term "application" can also be used to distinguish functions or code from infrastructure to support the execution of a component-based application, *e.g.* a *device worker* vs. an *application worker.*

**Application Specification (OAS)**
An XML document that describes the collection of *components* along with their interconnections and configuration properties in an OpenCPI *application.*

**Application Worker**
Implementation of a function used in an *application*, generally portable and hardware independent.

**Argument**
See *operation argument.*

**Artifact**
A file containing executable code for one or more *workers* for a specific *platform.*

**Authoring Model**
One of several ways of creating *component* implementations in a specific language using a specific API between the component and its execution environment. Existing models include RCC and HDL.

**Board Support Package**
A *project* that defines all items needed to enable OpenCPI on a given hardware and/or software *platform.* This includes, but is not limited to, *platform workers*, *device workers*, configuration of software cross-compilers, etc.

**Component**
Interface "contract" that is specified by a *component specification* and implemented by an *application worker.*

**Component Development Kit**
Set of tools, scripts, documents, and libraries used for developing *components* and *workers* in *projects.*

**Component Library**
Collection of *component specifications* and *workers* that can be built, exported, and installed to support *applications.*

**Component Specification (OCS)**

An XML document that describes both *configuration properties* and zero or more data interfaces (*protocol specifications*) of a *component*, establishing interface requirements for multiple implementations (*workers*) in **any** authoring model.

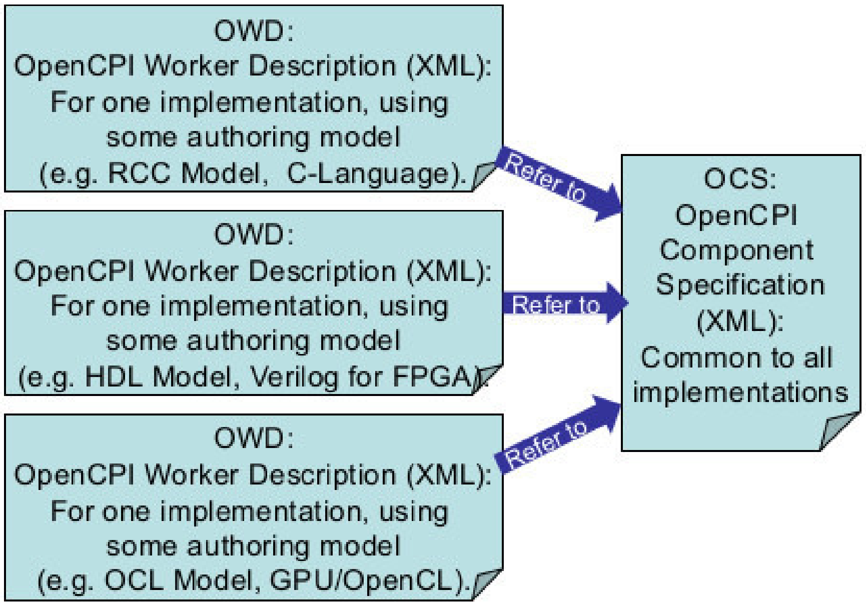
OWD:
OpenCPI Worker Description (XML):
For one implementation, using some authoring model
(e.g. RCC Model, C-Language).
OWD:
OpenCPI Worker Description (XML):
For one implementation, using some authoring model
(e.g. HDL Model, Verilog for FPGA).
OWD:
OpenCPI Worker Description (XML):
For one implementation, using some authoring model
(e.g. OCL Model, GPU/OpenCL).
Refer to
Refer to
Refer to
OCS:
OpenCPI Component Specification (XML):
Common to all implementations

Figure 1: Relationship between *OCS* and *OWD*s

**Configuration Properties**

See also *configuration property accessibility.*

Named values of a *worker* that may be read or written by *control software*. Their values indicate or control aspects of the *worker*'s operation. Reading and writing these property values may or may not have side effects on the operation of the *worker*. Configuration properties with side effects can be used for custom worker control. Each *worker* may have its own, possibly unique, set of configuration properties. They may include hardware resources such registers, memory, and state.

**Configuration Property Accessibility**

Declarations within an *OCS* or *OWD* that indicate **when** it is valid to read from or write to a *configuration property*. The various accessibility attributes (defined in the *Component Development Guide*) establish the rules in relation to the *worker*'s *lifecycle* or declare the property as fixed at build-time (see *Parameter*).

**Containers**

OpenCPI infrastructure element that "contains," manages, and executes a set of application *workers*. Logically, the container "surrounds" the workers, mediating all interactions between the *workers* and the rest of the system.

**Control Operations**

A fixed set of control operations that every *worker* has. The control aspect is a common control model that allows all workers to be managed without having to customize the management infrastructure software for each worker, while *configuration properties* are used to specialize components.

**Control Plane**

Control and configuration interfaces for runtime *lifecycle* control and configuration of *worker* instances at runtime.

**Control Software (AKA Control Application AKA Control Agent)**

The entity that is exercising control using the *ACI*. Encompasses the various aspects of how *control software*, usually running in a centralized host processing environment, can control *worker* instances at runtime via the *control plane*.

**Core**
The *project* containing the default *workers* and infrastructure VHDL for the framework to operate.

**Data Plane**
Data passing interfaces used for *workers* to consume/produce data from/to other workers in the *application* (of whatever *authoring model* in whatever *container*).

**Device Proxy**
A device proxy is a software *worker* (RCC/C++) that is specifically paired with a *device worker* in order to translate a higher level control interface for a class of devices into the lower level actions required on a specific device.

**Device Worker**
Special *worker* used for controlling hardware physically attached to an FPGA, *e.g.* a status LED.

**Hardware Description Language**
Refers to a specialized language used to program the structure design and operation of digital logic circuits. In OpenCPI, it is an *authoring model* using the VHDL language and is targeted at FPGAs. HDL *workers* should be developed according to the *HDL authoring model* and which is described in the "OpenCPI HDL Development Guide."

**HDL Assembly**
A fixed composition of HDL *workers* that can act as subset of a heterogeneous OpenCPI *application*.

**HDL Assembly Description (OHAD)**
The XML file that describes an *HDL assembly*.

**HDL Authoring Model**
The *authoring model* used by the HDL (VHDL-language) *workers*.

**Infrastructure**
Software/gateware is either *application* or infrastructure.

**isim**
The HDL simulator that Xilinx provides with their ISE software.

**Lifecycle Model**
The control states each *worker* may be in and *control operations* which generally change the state a worker is in, effecting a state transition:

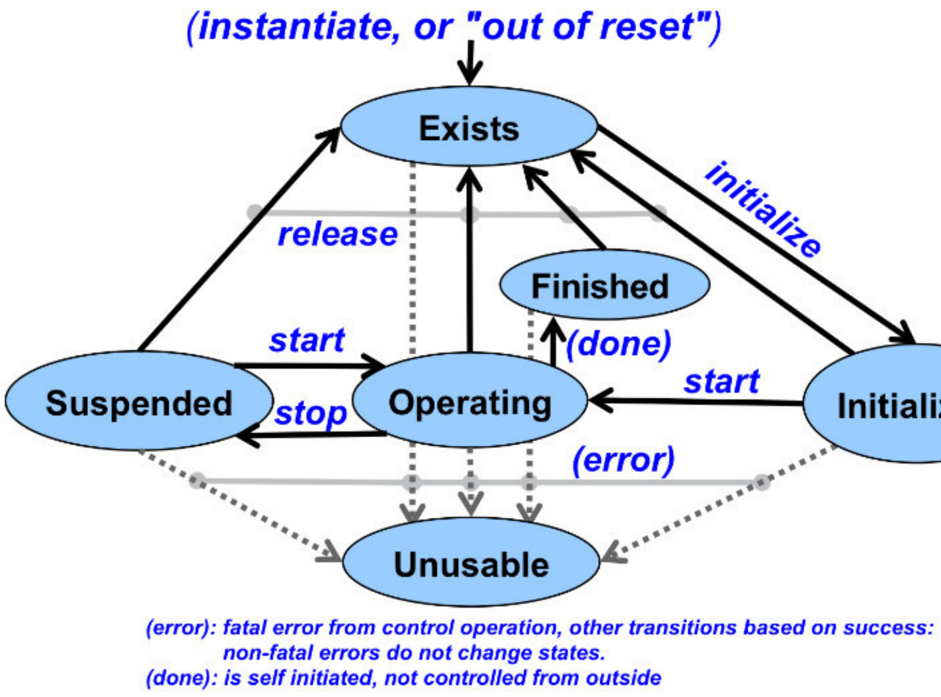
(instantiate, or "out of reset")
Exists
release
initialize
Finished
start
(done)
start
Suspended
stop
Operating
Initialized
(error)
Unusable
(error): fatal error from control operation, other transitions based on success:
non-fatal errors do not change states.
(done): is self initiated, not controlled from outside

Figure 2: The OpenCPI lifecycle model of **all** *workers*

**Library**
A conceptually-related set of *components* within a single location (often a *project*).

**OpCode**
See *operation code*.

**Operation Argument**
Payload data within a *protocol specification* whose type information is determined by the *operation code*.

**Operation Code**
Message type encapsulating zero or more *operation arguments* within a *protocol specification*.

**Parameter**
An immutable *configuration property* that is set at build time, allowing software compilers and hardware compilers to optimize accordingly.

**PCI (Peripheral Component Interconnect)**
Industry-standard local computer bus for attaching hardware devices.

**Port Readiness**
Indicates a *worker* has data available, input or output, that the *container* needs to act on. Input ports have available buffers when there is message data present that has not yet been consumed by the *worker*. Output ports are ready when buffers are available into which they may place new data.

**Platform**
A particular type of processing hardware and/or software that can host a *container* for executing OpenCPI *workers*.

**Platform Configuration**
The XML file that describes a unique configuration of a *platform*.

**Platform Worker**
A singleton *worker* that bootstraps the *platform* and *container*.

**Primitive**
HDL assets that are lower level than *workers* and may be used (and reused) as building blocks for HDL *workers*.

**Project**
Work area in which to develop OpenCPI *components*, *libraries*, *applications*, and other platform- and device-oriented assets.

**Project Registry**
A directory that contains references to *projects* in a development environment so they can be referenced by any *project* using that same *project registry*.

**Property**
See *Configuration Properties*.

**Protocol Specification (OPS)**
One or more XML files that describe the allowable data messages (*operation codes)* and payloads (*operation arguments*) that may flow between the ports of *components*.

**Protocol Summary**
The set of all summary attributes, whether inferred from the messages specified for the *protocol*, or specified directly as attributes of the protocol. Indicates the basic behavior of a port using a protocol. Can also be present when messages are specified, and can override the attributes inferred from the message specifications.

**RCC Authoring Model**
*Authoring model* used by the C or C++ language *workers* that execute on general purposes processors (GPPs). The "Resource Constrained" prefix indicates the limited set of library calls a worker should use; see the "OpenCPI RCC Development Guide" for more information.

**Registry**
See *Project Registry*.

**Run Condition**
When a *worker* has a combination of *port readiness* and/or some amount of time has passed. Determined by the worker's *container*.

**Run Method**
Non-blocking software method that is executed when a *worker*'s *run condition* is satisfied.

**Spec file**
Shorthand notation for *component specification* file.

**SpecProperty**
XML elements that add *worker*-specific attributes to the *configuration properties* already defined in the *component spec*.

**Worker**
Specific implementation of a *component specification* with the source code written according to an *authoring model*.

**Worker Description (OWD)**
The XML file describing the *worker* and references the *component spec* it is implementing. See Figure 1.

**XML**
Standardized markup language that defines a set of rules for encoding documents in a format which is both human- and machine-readable.

**xsim**
The HDL simulator that Xilinx provides with their Vivado software.

**Zero Length Message**
Data payload with no *operation arguments* present when a *protocol specification* allows such an *operation code* with no data fields.