Defining CORD

CORD® is an ONF open source project, and has also been accepted as a project in the Linux Foundation portfolio. This document describes the CORD ecosystem, with the goal of helping the community better understand how they can contribute to the CORD project.

Project Landscape

CORD is first and foremost an open source project. We characterize the landscape of that project as the following hierarchy of efforts:

- *CORD Vision* – The shared goal of the project is to promote the building of network edge facilities from cloud hardware and software technologies. More details about this Vision are presented in the next section, and while we can expect this Vision to evolve over time, it does establish the common ground for all CORD-related efforts.
- *CORD Architecture* – A collection of interfaces and models that define a particular way to achieve the CORD Vision. Today there is one Reference Architecture (defined later in this document), but other Architectures are possible.
- *CORD Implementation* – An integrated collection of hardware and software components that implement a CORD Architecture. There is one open reference implementation today that implements the Reference Architecture (defined later in this document), but other implementations are possible.
- *CORD Solution* – A specific configuration/customization of a CORD Implementation targeted at a particular use case. Several example Solutions are available in today's Reference Implementation, each centered on a different access technology (e.g., R-CORD, M-CORD, E-CORD), but this is a short-term situation. A much richer collection of Solutions, including ones that support multiple access technologies, are expected in the near future.
- *CORD Platform* – The subset of a CORD Implementation that is common to all CORD Solutions. The components that make up the Platform in today's Reference Implementation (the Reference Platform) are described later in this document. It is also possible to give an architectural definition of a CORD Platform in terms of the models and interfaces of a CORD Architecture.
- *CORD Distribution* – A packaging and release of implementation components, tailored for a particular user community. Today there is one Distribution—it is exactly the same as the Reference Implementation and includes both the Reference Platform and a collection of Reference Solutions—but multiple Distributions are possible, and they need not be mutually exclusive. For example, different organizations could prepare Distributions that include the same Reference Platform, but different combinations of Solutions.

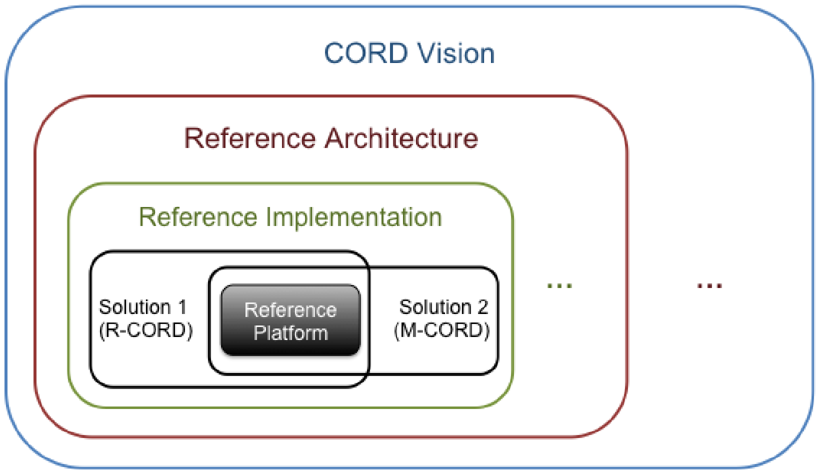

Figure 1. CORD Project Landscape.

The richness of the project hierarchy (as depicted in Figure 1) supports a wide range of opportunities to participate and contribute. For example, it is possible to define alternative Architectures that are consistent with the CORD Vision, and multiple Implementations of today's Reference Architecture are welcome and encouraged. As another example, multiple Solutions and Distributions built around today's Reference Platform are already starting to emerge.

While there is value in diversity, there is also value in the community converging around a common set of definitions. For example, alignment on a common Architecture increases the ability to create interoperable components, and alignment with the Reference Implementation broadens the ability to build on each other's open source components.

The rest of this document describes the CORD Vision in more detail, and also gives some specifics about today's Reference Architecture and Reference Implementation, both of which are managed by the CORD Technical Steering Team (TST) [1]. In doing so, there are two points to keep in mind.

First, this document focuses on definitions. It does not prescribe a certification process that might be used to establish whether a given implementation of CORD complies with these definitions. For the sake of exposition, however, it is useful to talk in general terms about how well a given implementation meets these definitions. In such situations, we say an implementation "aligns" with CORD (as opposed to, is "compliant" with CORD), and we describe the means by which this alignment is established (without claiming the high bar of certification).

Second, the CORD Reference Architecture and Reference Implementation are closely linked today. As one would expect, the Reference Implementation is a realization of the Architecture, but it is also the case that the Reference Implementation is the source of the architectural definition. This cyclic dependency is rooted in the practice of basing standards on running code. If at some point in the future there are multiple Implementations of the CORD Reference Architecture, it will make sense to revisit this preferential status for the open Reference Implementation.

Refer to this introductory video to CORD here:

Vision

While much of this document focuses on the CORD Architecture and Reference Implementation, there is a large community working towards the common goal of introducing cloud technologies into the edge of operator networks (e.g., Telco Central Offices). Efforts in this larger ecosystem are aligned with the CORD Vision as long as they are consistent with CORD's Architectural Requirements [2]. Some efforts might be more narrowly focused, and so do not address all the requirements, but five principles stand out as a requirement for any claim to be aligned with the CORD Vision:

1. Is built around commodity servers and software controlled switches, and to the extent possible, leverages merchant silicon.
2. Enables disaggregation, and is not restricted to running bundled legacy VNFs in virtual machines.
3. Leverages SDN and is able to run fully programmable control applications, both to interconnect the virtual and physical elements and as a source of innovative services.
4. Is extensible by design, with a common core (platform) that can be configured to include different combinations of access technologies and services.
5. Adopts best practices in building, composing, and operating scalable multi-tenant cloud services.

As noted above, while any effort that is consistent with this vision moves the industry forward, there is significant value in doing so in a way that also aligns with the CORD Reference Architecture (the ability to create interoperable components), and even more so, aligns with the CORD Reference Implementation (the ability to build on each other's open source components). The following sections call out different ways to contribute to both of these.

Reference Architecture

The CORD Reference Architecture follows from a set of requirements [2], and in particular, meets all of the principles underlying the CORD Vision. Today, the architecture is specified by the public API for the CORD reference implementation. As of the Dangerous-Addition release, this API is auto-generated from a set of declarative models, and as a consequence, these models constitute the definitive specification of the CORD Reference Architecture.

These models not only define the API, but because they codify invariants about the internal behavior of CORD, they also prescribe structure on the implementation (i.e., they define abstract building block components, impose interfaces on those components, and establish relationships among those components). For example, Solutions are defined by a Service Graph; a Service Graph consists of a set of Services and Dependencies between Services; a Service consists of a Controller and one or more Slices; a Slice is a logical resource container that includes a set of Instances and a set of Networks, and so on. As a consequence, these models also allows us to define the CORD Platform in architectural terms:

- *CORD Reference Platform (Architectural Definition)* – An implementation of CORD that faithfully supports the API and core models defined by the CORD Reference Architecture.

Because CORD is designed to be extensible, and it is extended by on-boarding models for some set of services, the "core" qualifier in this definition is important. The core model definitions for the latest release can be found in the source code [3].

This begs the question of what it means to "faithfully implement the API." In general, this is a question of certification, but pragmatically, an obvious test would be for a candidate implementation of the CORD Reference Platform to pass the same QA tests as are used to qualify a release of the Reference Implementation. For the purpose of this document, such an implementation is said to align with CORD.

Note that while these QA tests could evolve into a more complete certification process, in general, the goal is to construct a suite of tests that demonstrate that a candidate implementation is able to:

- On-board a representative collection of services, and inter-service dependencies
- Provision, configure, and control individual services and service instances
- Deliver the desired functionality, on a service-by-service basis

The Requirements document provides guidance in the breadth of what a functional test suite should cover. Also note that beyond this functional minimum, one might also want to test various other properties of an implementation, such as performance, scalability, availability, and security.

Returning to the cyclic dependency between the Reference Architecture and the Reference Implementation, our expectation is that the architectural definition will eventually be decoupled from latest Reference Implementation. The API and model definitions will become more self-contained, and will evolve in response experience with multiple implementations.

Reference Implementation

The CORD Reference Implementation is a realization of the CORD Reference Architecture, packaged as a combination of hardware and software that has been approved for release by the CORD Technical Steering Team (TST) [1]. This includes a Bill-of-Material for validated hardware, instructions for how to assemble this hardware into a CORD POD, and a software installation process that builds a prescribed configuration and deploys it on the target hardware.

It is most accurate to think of the Reference Implementation as having two parts. The first is CORD Reference Platform. The second is a package of one or more Reference Solutions that extend (build upon) the Platform to address various operator needs. Today, the Reference Solutions are centered on different access technologies—e.g., M-CORD, E-CORD, R-CORD—but this is not a requirement. We expect a much richer collection of Solutions in the future, including ones that support multiple access technologies.

In the context of the TST-sanctioned Reference Implementation, each Solution builds on (extends) the Platform as specified by:

- *Service Profile* – A software specification of the set of Services to load onto the Platform.
- *Hardware POD* – A bill-of-material and wiring diagram for the target hardware.

The Solutions included in each release of the Reference Implementation are tagged according to their level of maturity. The latest information about these configurations is maintained on the CORD wiki [4]. Currently only two levels are defined:

- *Official* – Passes QA tests and approved by the TST for release
- *Development* – Not yet passed QA tests nor approved by the TST for release

where additional status levels (e.g., Deprecated, Experimental) might be added over time. The rationale behind including Solutions that have not yet been fully vetted by the TST is to make software available to the community as early as possible. The important point, from a definitional standpoint, is that it is only those Solutions marked as Official that we consider in the following hierarchy of definitions:

- *CORD Reference Distribution* – A packaging of the CORD Reference Platform and a collection of CORD Reference Solutions approved by the CORD TST. The following definitions assume that context.

- *CORD Reference Solution* – A combination Official Service Profile and target Hardware POD tested and included in a release of the CORD Reference Implementation.

- *CORD Reference Service* – Any Service included in the Service Profile for one or more of the CORD Reference Solutions.

- *CORD Reference Hardware* – Any hardware device included in a Hardware POD for one or more of CORD Reference Solutions.

Viewing the reference implementation as a CORD Distribution (in the sense of a Linux distribution) re-enforces the point that there could be multiple CORD Distributions, each including its own collection of Solutions, but all building on the common CORD Reference Platform included in the reference implementation. It also suggests that different Distributions could adopt different strategies for qualifying a particular set of Solutions, as long as those solutions were distributed with the reference implementation of the CORD Platform, analogous to a Linux Distribution minimally including the Linux Kernel.

As one simple example of how a secondary distribution could build upon the TST-sanctioned Distribution, any Solution derived from a Solution included in the TST-sanctioned release by adding/removing CORD Services or adding/removing CORD Hardware.

Discussion

This section discusses several tangential points raised by these definitions. For simplicity, we drop the “Reference” qualifier, but the comments that follow are offered in the context of today’s Reference Architecture and Implementation.

Services and VNFs

This document often uses the terms “Service” and “VNF” interchangeably, but a Service is a VNF that has been on-boarded into CORD. This means a VNF is aligned with CORD if it is part of (included in) a CORD Service. Note that although it is the common/expected case, there is not necessarily a one-to-one correspondence between VNFs and Services. For example, it is possible to define a CORD Service that effectively combines multiple VNFs. It is also the case that some Services are implemented in the underlying virtual/physical switches and managed by a control application running on ONOS. Whether one considers such an application a VNF depends on your definition of a VNF.

Service Dependencies

A Service Profile defines a graph of dependencies among a collection of Services; arbitrary combinations of Services are not necessarily meaningful. These dependencies, in turn, qualify how widely or narrowly a given Service can be used (e.g., Service A can only be used if Service B is also present in the Profile). Of particular note, VNFs typically depend on a specific virtualization technology (e.g., KVM, Docker), which in CORD is expressed as a dependency on some underlying infrastructure service (e.g., OpenStack). The CORD architecture is sufficiently general to support multiple infrastructure services, where a given VNF is considered functional with respect one or more of the infrastructure services aligned with CORD.

Platform Components

While we have an architectural definition of the CORD Platform, a common question is what constitutes the platform in practice (e.g., does it include OpenStack)? Practically speaking, the current reference implementation includes a core set of components that can fairly be called the Reference Platform. All CORD Solutions available today run on this reference platform, which includes XOS, ONOS, OpenStack, Docker, VTN (an ONOS app that implements overlay virtual networks), and Fabric (an ONOS app that manages the switching fabric).

Said another way, CORD is designed around a minimal kernel that can be extended by adding services, but a certain set of services must be loaded into this kernel to get a minimally viable system. This minimally viable system is the reference platform, on top of which various CORD Solutions are built.

CORD Distributions

As mentioned earlier in this document, the TST-approved Reference Implementation could eventually be one of many possible CORD Distributions. Presumably, one could then define a “CORD Distribution” (either fully or partially open source) that includes one or more CORD Solutions (in this case it would be a “Distribution Solution” as opposed to the “Reference Solution”). Such a distribution would likely be qualified outside the current TST-approved release process, but it would need to align with at least the CORD Platform definition, analogous to a Linux Distribution minimally including the Linux Kernel.

In general, there are at least two ways the CORD ecosystem could evolve beyond a single Reference Implementation. One is for multiple distributions, sharing a common reference implementation of the CORD Platform, packaging and distributing different combinations of CORD Solutions. A second is that there could be additional implementations of the CORD Platform, all aligned with the CORD Reference Architecture, but each of which could be bundled with its own set of Solutions.

These two paths for growing the ecosystem reveal a subtle aspect of the definitions presented in this document. CORD Platforms are defined in terms of the CORD Architecture (which today implies one Reference Platform, but others are possible), while releases/distributions of CORD Reference Solutions are defined relative to the Reference Platform, along with the release processes of some organization (which today implies the TST-approved set of Solutions, but other organizations could prepare their own distributions of Solutions, but still bundle them with the TST-approved Reference Platform).

Open vs Proprietary Services

It is not the case that source code for all Services included in a Service Profile will be released as part of the open reference implementation. This is because we expect services built using proprietary VNFs will be included in one or more of the Official Service Profiles. (Any models needed to on-board the service into one of the solutions will be included in the open source release, but not necessarily source code for the VNF itself.) As a consequence, there will be CORD Services that are not open and fully included in the reference implementation. An important restriction is that it includes at least one open infrastructure service be included in a reference implementation, but it is also possible to build a CORD Platform that includes proprietary infrastructure services (or proprietary enhancements to open source infrastructure services).

Although beyond the scope of this document, we assume the TST is granted an evaluation license for any Service/VNF that is a candidate for inclusion in the reference implementation, and that others could get a similar license for a trial of the reference implementation.

TST Release Approval

Because many of the definitions presented in this document are anchored by the Reference Implementation, the question of how the CORD TST decides what to include in a given release is an important one. The process is still evolving, but the answer includes both release planning (i.e., deciding how to prioritize development for inclusion in the next release) and QA testing (deciding what set of tests the solutions included in a release must pass). Information about the CORD roadmap and QA testing are available on the CORD wiki [5,6]. Note that the primary objective of QA testing is to produce a viable release; it is not currently intended to serve as a certification process for Services and Solutions.

CORD-Based Solutions

Both the CORD Reference Architecture and the CORD Reference Implementation define an integrated whole, with the latter built from a collection of self-contained software components, many with their own well-defined APIs. An important aspect of the CORD Platform is how it integrates these components under a coherent set of model definitions and unified Northbound interface.

These individual components can often be used in isolation, apart from the Reference Implementation as a whole. A system built in this way would certainly align with the CORD Vision, but more generally, could be integrated in different ways to form alternative CORD Architectures.

Appendix: Glossary

This appendix collects together the definitions presented throughout this document.

CORD Vision – An approach to building network edge facilities from cloud hardware and software technologies. Narrowly defined around a set of requirements [2], the Vision adopts a set of principles revolving around commodity hardware, disaggregation, SDN-based control, extensibility, and best practices in scalable cloud services.

CORD Reference Architecture – Currently specified by the public API for the CORD Reference Implementation, which is in turn auto-generated from a set of declarative models. Designed to satisfy a set of requirements [2] that are aligned with the CORD Vision.

CORD Reference Implementation – The combination of hardware and software that has been approved for release by the CORD Technical Steering Team (TST). Each release includes software that is still under development, so the reference implementation definitionally includes only those components qualified as Official.

CORD Distribution – A packaging of one or more CORD Solutions, bundled with the CORD Platform included in the Reference Implementation. The CORD TST has a process for defining a set of Solutions to include in each release, making the reference implementation the only distribution available today. Other distributions that include different combinations of Solutions, but leverage the same reference implementation of the CORD Platform, are also possible.

CORD Platform – The kernel of an implementation of CORD that faithfully supports the public API and core models defined by the CORD Architecture [3]. Today's CORD Reference Platform includes the following software components: XOS, ONOS, OpenStack, Docker, VTN (an ONOS app that implements overlay virtual networks), and Fabric (an ONOS app that manages the switching fabric).

CORD Solution – In the reference implementation, defined by a combination of CORD Platform, Service Profile, and target Hardware POD tested and included in a CORD release. More generally, Solutions could also be built on other implementations of the CORD Platform.

CORD Service – Any Service included in the Service Profile for one or more CORD Solutions included in the Reference Implementation.

CORD Hardware – Any hardware device included in a Hardware POD for one or more CORD Solutions included in the Reference Implementation.

Service Profile – A software specification of the set of services to load onto the CORD Platform. Currently specified as Official by the TST for the reference implementation [4], and qualified by a set of QA tests [5].

Hardware POD – A bill-of-material and wiring diagram for the target hardware that runs a particular Service Profile. Currently specified as Official by the TST for the reference implementation [4], and qualified by a set of QA tests [5].

CORD Technical Steering Team (TST) – A technical governing body established under the governance of CORD, operating under the auspices of the Linux Foundation [1]. The TST is responsible for the overall technical direction of the CORD project, with includes approving releases of the CORD Reference Implementation.

References

1. CORD Governance. https://wiki.opencord.org/display/CORD/CORD+Governance.
2. Architectural Requirements. https://wiki.opencord.org/display/CORD/Requirements.
3. Core Model Definitions. https://github.com/opencord/xos.
4. CORD Service Inventory. https://wiki.opencord.org/display/CORD/Service+Inventory.
5. CORD QA Tests. https://wiki.opencord.org/display/CORD/System+Test.
6. CORD Roadmap. https://wiki.opencord.org/display/CORD/Roadmap.