CORD Requirements

CORD is a reference implementation of a service delivery platform for network operators. It is a fully integrated system – sufficiently complete to support field trials – built from open source components. Being an integrated system is an essential aspect of CORD, as our goal is to hold the project to a higher bar than a collection of loosely related projects listed under a common banner. Being open is another essential aspect of CORD. This does not imply that all modules configured into CORD need be open source (e.g., proprietary services can run on the platform), but it is important that both the CORD platform be open source and CORD as a whole be populated with enough open source services to demonstrate the full breadth of its capabilities.

In addition to being open and integrated, CORD is designed to satisfy the following five requirements. The first three follow directly from the goal of of bringing cloud economies and agility to the Telco setting. The last two are about applying best practices in system design to this specific domain.

Economies of Commodity Hardware

CORD must run on commodity servers and white-box switches, and to the extent possible, leverage merchant silicon; it should not depend on proprietary or purpose-built hardware to achieve high performance. That CORD leverages commodity hardware follows directly from its goal of supporting the economics of cloud infrastructure. And by implication, the software running on that commodity hardware must deliver the same performance and reliability as today's purpose-built hardware.

To this end, CORD provides mechanisms to virtualize commodity servers and control white-box switches, along with several open source exemplars of how software can leverage these mechanisms to achieve high performance and reliability.

Enable Innovative Services

CORD must support a wide-range of services; it is not limited to access services, nor should it unnecessarily constrain how services are implemented. Specifically, CORD must support services drawn from across the following four dimensions: (1) both access services (e.g., Fiber-to-the-Home) and conventional cloud services (SaaS); (2) both services implemented in the data plane (NFV) and services implemented in the control plane (SDN); (3) both trusted operator-provided services and untrusted third-party services; and (4) both bundled legacy services and disaggregated greenfield services. Moreover, CORD must provide an agile framework for managing a set of services without regard for how the individual services achieve isolation, scale, performance, and high availability.

To this end, CORD defines a unifying service abstraction (and other supportive models) that encompasses the full range of service designs, along with mechanisms that map those abstractions onto virtualized commodity hardware.

Extensible and Controllable

CORD is a configurable platform; it is not a point-solution. It must provide a means for the operator to specify the desired collection of services, as well a full OAM capability for those services. This includes the ability to configure CORD for different markets and access technologies: Residential, Enterprise, and Mobile. It must also provide a means to provision and parameterize those services to meet the operator's operational and business objectives.

To this end, CORD defines a Northbound Interface (NBI) that allows operators to configure, control, and extend a CORD deployment, and does so in a way that cleanly separates policy and mechanism.

Multi-Domain Security

CORD must account for multiple domains of trust; it is not sufficient to only distinguish between CORD operators and CORD users. This includes a wide range of intermediate roles, including global operators, site-specific operators, service operators, service developers, service tenants (other internal services) and service subscribers (external principals).

To this end, CORD adopts best practices in secure system design: it minimizes the trusted code base, it provides mechanisms to mediate trust, and it provides mechanisms that support the principle of least privilege.

Operational Robustness

CORD must account for partial and intermediate failures and incremental upgrades; it is not acceptable to ignore the operational realities of building a system by integrating multiple, independently developed and deployed software components. Without holding the reference implementation to the high bar of "production ready," CORD must be architected for zero-touch: to account for the possibility that the operational behavior of the system is not always in sync with the target state of the system, and to automatically steer the system to the correct functioning state.

To this end, CORD employs best practices in scalable cloud services: to treat error states as expected in a production system, to provide built-in mechanisms to automatically recover from failures, and to support incremental hardware and software upgrades without service disruption.