



OPEN
Compute Project

OCP Foundation Project

OCP Contribution Taxonomy

DRAFT

Informational
Effective June 10, 2026

Author: OCP Foundation Staff

Table of Contents

1	Contribution Taxonomy Overview	5
1.1	Definition & Usage	6
2	Requirements & Strategy	9
3	Standards & Guidance	11
4	Design & Implementation	15
5	Validation & Operations	18
6	Maintenance	19
7	Training	20
8	Summary	21
	Appendices	22
A	Contribution Decision flow	22
B	Contribution Evolution Examples	23

List of Figures

1	Community Goals	5
2	Mapping to contribution	7
3	Specification mapping.....	12
4	Mapping to contribution, alternate view	21
5	Choosing a Contribution	22
6	Example: Requirements Document to Specification	23
7	Example: White Paper to Source code	24

List of Tables

1	Tools Examples	16
2	Usages	21

Revision History

Revision	Date	Author(s)	Description
DRAFT	20260605	Russ Wunderlich (OCP)	Initial draft for Community review

1 Contribution Taxonomy Overview

The following defines the taxonomy of Open Compute Project (OCP) contributions providing overall definition of contribution categories along with their affinities and associated details. This taxonomy is invariant to the source of the contribution.

The OCP Community is broad and serves many diverse purposes. To ensure consistency and alignment, three high-level goals guide how contributions are defined and utilized:

1. Introduce new technology and direction to the Community
New technology introduction is satisfied by shaping strategy and defining requirements as well as establishing standards and industry guidance.
2. Enable a full ecosystem
Enabling a full ecosystem is satisfied by establishing standards and industry guidance as well as inform design and implementation
3. Support ongoing technology evolution
Technology Evolution is satisfied by inform design and implementation as well as support validation and operational capabilities.

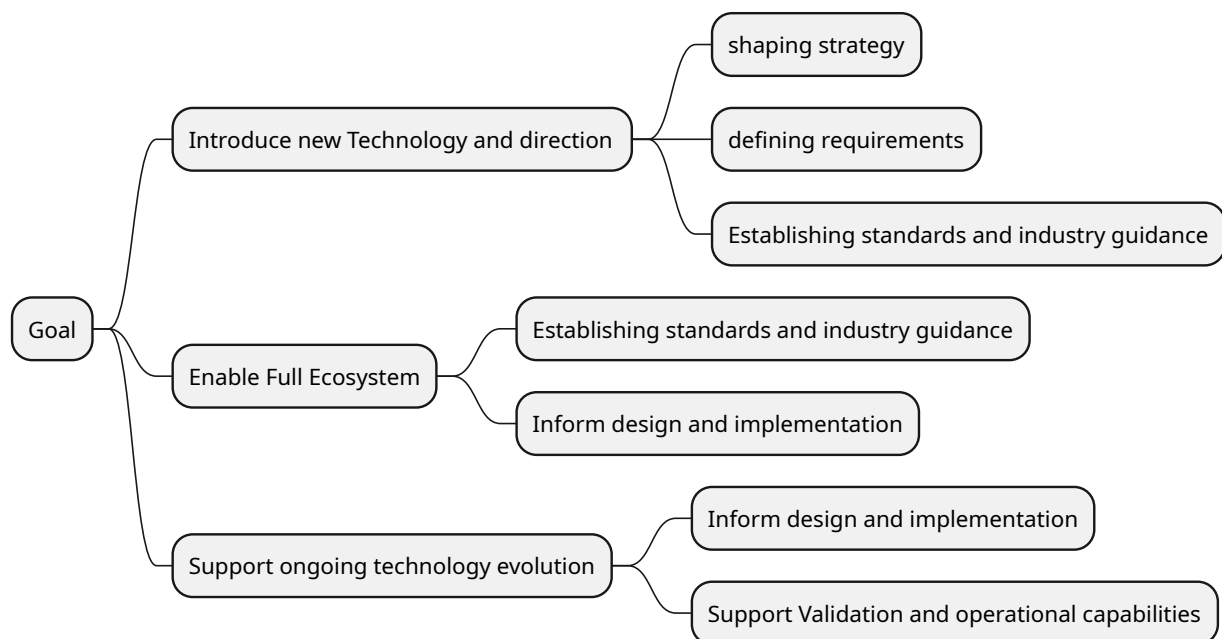


Figure 1: Community Goals

This mapping is not intended to be restrictive. Rather, it ensures that each definition and usage of OCP contributions remains clearly aligned with these goals.

1.1 Definition & Usage

Six usages are defined for OCP Contributions spanning the full development lifecycle from concept to operational deployment, maintenance, and education:

1. Shape strategy and define requirements
Provide insights that influence strategic direction and foundational needs
2. Establish standards and industry guidance
Create shared frameworks, best practices, and specifications adopted across the ecosystem
3. Inform design and implementation
Contribute technical detail that supports architecture, engineering, and solution development
4. Support Validation and operational capabilities
Enable testing, validation, and real-world operational readiness
5. Maintenance
Ongoing support of a contribution
6. Training
Educating the Community

These usage categories help guide the Community in developing, classifying, and applying OCP contributions effectively and consistently.

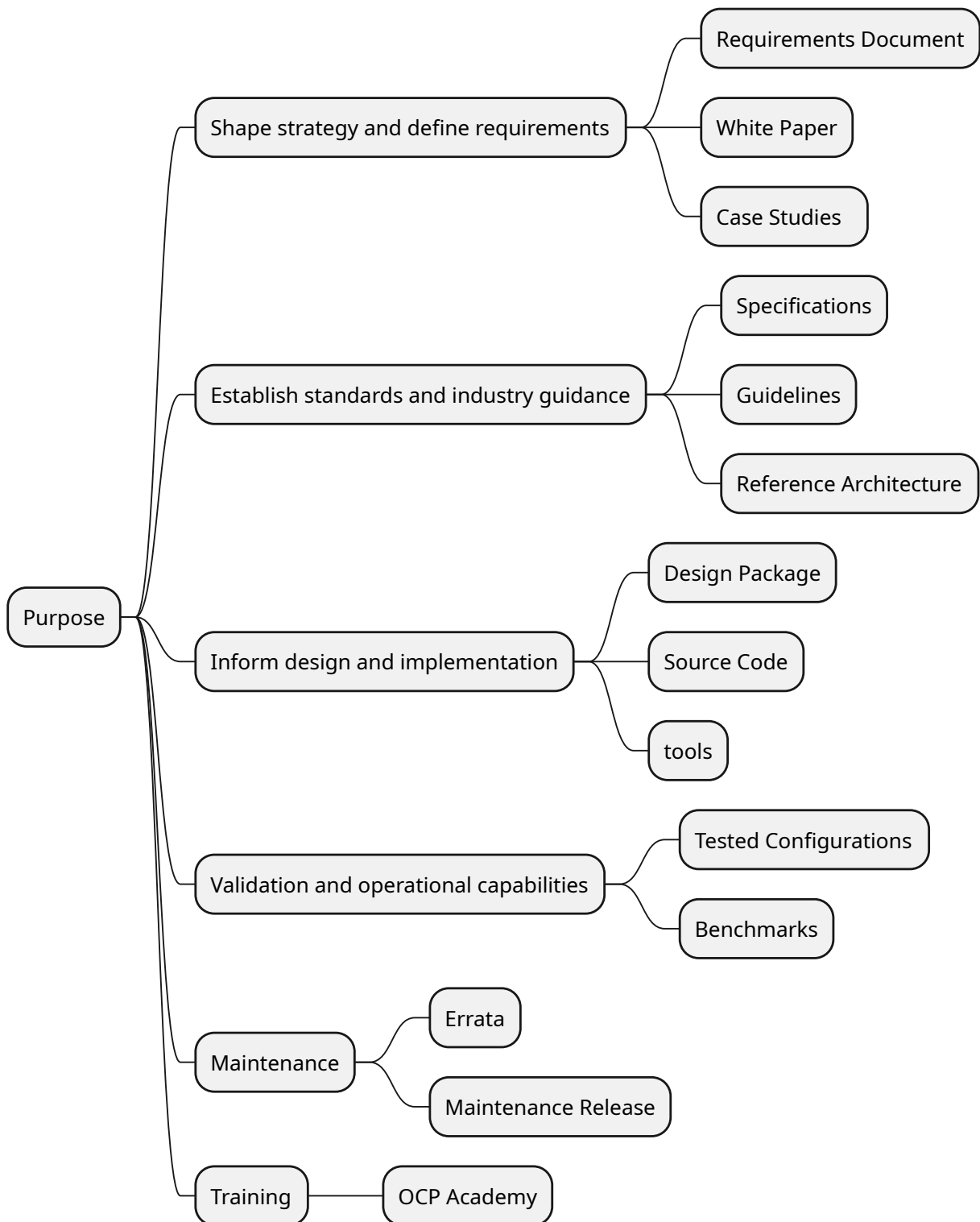


Figure 2: Mapping to contribution

38 The following sections map usage scenarios to the corresponding OCP contributions and
39 define each contribution type.
40 Each section includes a description of the contribution type, the prerequisites for
41 submission, the allowed contribution formats (both the OCP provided templates and the
42 source materials), and any permitted collateral or supporting documents.
43 Contributions may be submitted in source and rendered formats or as a GitHub link. Any
44 Markdown contribution must be submitted as a GitHub link.

45

2 Requirements & Strategy

Three key contribution types help shape industry direction and articulate emerging needs:

- Requirements Document
- White Paper
- Case Studies

Requirements Document: Authoritative statement of needs, constraints, and acceptance criteria.

A requirements document is an authoritative statement of a system's needs, constraints, and acceptance criteria that describes what a solution must achieve (functional and non-functional requirements) without prescribing how to implement it.

Examples:

1. Product Requirements Document (PRD): defines the purpose, features, functionalities, and expected behavior of a product
2. Technical Requirements Document (TRD): captures all technical specifications, functionalities, performance standards, design constraints, interfaces, and other technical details for a product or solution
3. Business Requirements Document (BRD): outlines the business objectives, needs, and expectations for a proposed project

Contribution Prerequisites: None

Template & source document: Document Template (OCP-aligned Markdown, .docx, Google Docs)

Allowable supporting documents: Supporting data packages of various formats (spreadsheet, domain tool, etc.)

White Paper: Conceptual, strategic, or thought leadership content explaining approach or rationale.

A white paper is a detailed, authoritative document that examines a specific problem, issue, or innovation and presents a well-researched, evidence-based solution or perspective. It is designed to inform, influence, or educate readers about a particular topic through expert insight, data, and analysis. White Papers can provide best practices, highlight research findings, or recommend approaches.

Contribution Prerequisites: None

Template & source document: Document Template (OCP-aligned Markdown, .docx, Google Docs)

Allowable supporting documents allowed: Supporting data packages of various formats (spreadsheet, domain tool, etc.)

Case Studies: Evidence of outcomes and lessons learned from real implementations.

Case studies provide evidence of outcomes and lessons learned from real implementations, offering a narrative that documents a real-world problem, the actions taken to address it, and the measurable results achieved. They demonstrate how a product, service, or

strategy effectively solved a customer’s challenge, delivering concrete, real-life validation of its value. Developed to inform, influence, or educate, case studies may highlight best practices, showcase research findings, or propose new directions. They serve as powerful tools across marketing, sales, engineering, and other organizational teams by supporting customer discovery, reinforcing relationships, establishing thought leadership, proving solution effectiveness, and inspiring Community engagement.

A case study may appear in various media formats, including documents or video, and typically sets up a problem statement and demonstrates how a customer lacked, or used a product or solution to address the problem. Any video submission must include a document component establishing references, outlining potential future work, and acknowledging collaborators.

Contribution Prerequisites: None

Template & source document: Document Template (OCP-aligned Markdown, .docx, Google Docs), Videos (MP4 or other suitable source; external links are not acceptable)

Allowable supporting documents allowed: Supporting data packages of various formats (spreadsheet, domain tool, etc.)

101

3 Standards & Guidance

Three key contribution types create shared frameworks, best practices, and standards adopted across the ecosystem:

- Specifications
- Guidelines
- Reference Architecture

Specifications: Formal, testable definitions of behavior, interfaces, or protocols (prescriptive).

A specification is a formal document that defines the requirements, behaviors, interfaces, constraints, and intended characteristics of a system, component, product, or process. It establishes a clear and authoritative description of what must be built or achieved, providing the measurable criteria and technical information necessary for design, implementation, verification, and interoperability. Specifications serve as a common reference to ensure consistency, alignment, and repeatability across teams, tools, and lifecycle stages.

Examples: API specs, electrical/mechanical specs, data schemas.

Three specification types are established to cover architectural, design, and product-level requirements for hardware and software systems. Each specification is classified according to its principal purpose but may incorporate elements from other specification categories as needed to meet that purpose. As a result, a single specification may encompass architectural, design, and product-related content when required to achieve its defined goals.

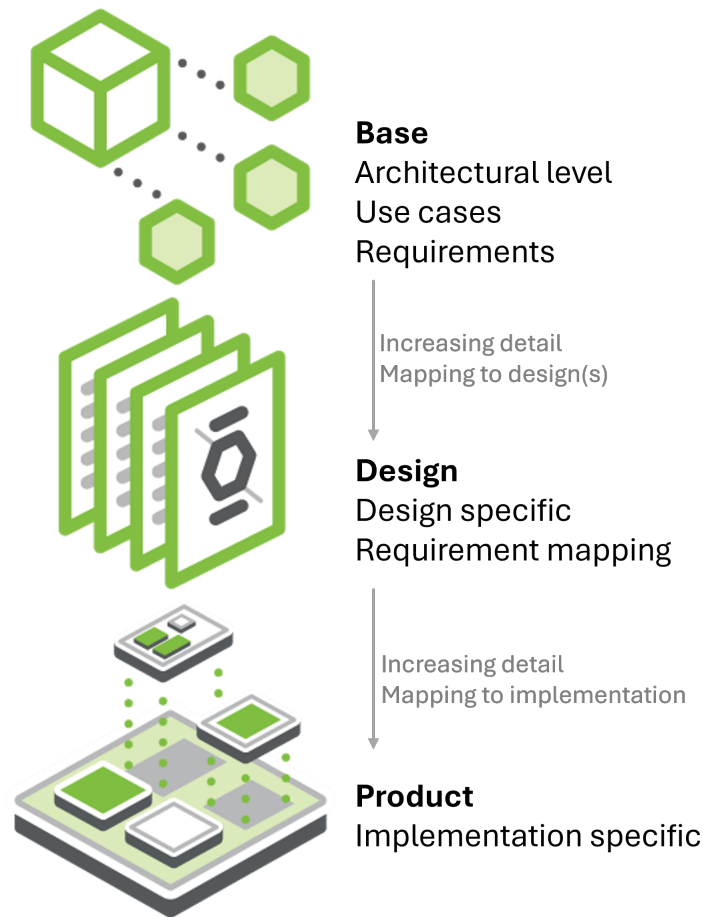


Figure 3: Specification mapping

• Base Specification

A Base Specification defines the architectural framework and system-level requirements that enable broad alignment and interoperability across hardware, software, and firmware components. It establishes the foundational interfaces, operational parameters, and design constraints upon which subsequent Design and Product Specifications are built.

The Base Specification describes the overall system, subsystem, or functional architecture at a reusable and generic level. It includes the essential elements needed to understand the system's structure and behavior, such as:

- Design requirements
- System and subsystem architecture
- Database or interface-level design concepts
- Descriptions of systems, services, platforms, and relationships among modules
- High-level test and verification concepts

The Base Specification is market-driven, broad in scope, and typically light on IP-specific detail. It must remain sufficiently generic to ensure compatibility across multiple implementations, generations, and derivative specifications. Like an API contract in software development, the Base Specification provides a stable reference point that subsequent Design Specifications can extend while maintaining fundamental interoperability.

Contribution Prerequisites: None

Template & source document: Specification Template (OCP-aligned Markdown, .docx, Google Docs)

Allowable supporting documents allowed: None

• Design Specification

A Design Specification refines and expands the architectural requirements defined in the Base Specification. It captures customer, system, and use-case requirements with enough engineering detail for developers to begin realizing the solution. It defines the intended role, boundaries, and functional objectives of the design and includes architectural diagrams, subsystem descriptions, integration requirements, and other detailed engineering information.

The Design Specification provides a technical representation of the system, subsystem, or function being developed. It specifies the data structures, algorithms, component behaviors, and interface details required to guide implementation and ensure that independently developed components interoperate correctly.

When the deliverable includes or is an Application Programming Interface (API), the Design Specification encompasses the full API definition. This includes functional behavior, supported data types, expected request-response patterns, and the underlying design philosophy. An API specification serves as a contract—expressed through documentation and formal interface definitions—that enables authorized users and software systems to reliably access the resource through structured requests.

Multiple Design Specifications may derive from a single Base Specification, enabling common interfaces while supporting generational evolution and product differentiation.

Contribution Prerequisites: None

Template & source document: Specification Template (OCP-aligned Markdown, .docx, Google Docs)

Allowable supporting documents allowed: Supporting data packages of various formats (domain tool (STEP, BRD, ...), Spreadsheet (BOM ...), etc.)

• Product Specification

A Product Specification defines the complete, manufacturing-ready requirements necessary to build a product consistently and reliably from an associated Design Specification. It provides the detailed mechanical, electrical, thermal, software, and interface information required for multiple manufacturers to produce functionally identical and interoperable products. Any significant variation requires a separate Product Specification.

For software-based systems or software components within a product, the Product Specification includes a fully detailed design of the software system, covering explicit descriptions of data structures, algorithms, interfaces, state behavior, and other architectural or functional elements. It may also include test definitions, pseudocode, sequence flows, or other technical representations necessary to ensure conformance across independently developed software components.

Overall, the Product Specification reduces design variability, ensures repeatability, and provides the complete level of detail—across hardware, software, and firmware—required to assemble, verify, and manufacture the final product based on the associated Design Specification and, ultimately, the Base Specification.

Contribution Prerequisites: None

Template & source document: Specification Template (OCP-aligned Markdown, .docx, Google Docs)

Allowable supporting documents allowed: Supporting data packages of various formats (domain tool (STEP, IPC-2581, Gerber, ...), Spreadsheet (BOM ...), etc.)

Guidelines: Recommended practices and patterns (advisory).

Guidelines are advisory documents that outline best practices, recommended assessments, and standardized procedures to support consistent, high-quality decision-making and implementation across projects or systems. They provide structured, experience-based direction without imposing strict requirements, helping teams align on effective methods while maintaining flexibility in execution.

Contribution Prerequisites: None

Template & source document: Document Template (OCP-aligned Markdown, .docx, Google Docs)

Allowable supporting documents allowed: Supporting data packages of various formats (spreadsheet, domain tool, etc.)

Reference Architecture: Authoritative system topology and integration patterns; a “blueprint”.

A reference architecture is a documented, structured template solution that defines recommended components, patterns, and integrations for a particular technical domain, providing a reusable blueprint that guides the design and implementation of systems. A reference architecture typically consists of two or more unique, tested configurations that demonstrate validated system behavior and proven design patterns. Contributors are required to provide product test documentation, certification information, and complete configuration details, and warrant that the reference architecture functions as documented. This ensures that the architecture is both credible and repeatable, offering a trusted foundation for deploying interoperable, high-performance, and resilient solutions

Contribution Prerequisites: None

Template & source document: Document Template (OCP-aligned Markdown, .docx, Google Docs)

Allowable supporting documents allowed: Supporting data packages of various formats (spreadsheet, domain tool, etc.)

4 Design & Implementation

Three key contribution types provide technical detail and support for solution development:

- Design Package
- Source Code
- tools

Design Packages: Design artifacts such as diagrams, models, manufacturing files, and BOMs.

Design Packages, also referred to as “Complete Production Files” or “Design Files,” comprise the full set of engineering artifacts, documentation, and data required to enable a person of ordinary technical skill to manufacture the Product or to modify its design. Design packages accompany a specification and are optional except when seeking an OCP Accepted™ product recognition, in which case they are mandatory.

Note: this design package requirement is removed with the upcoming marketplace definition change.

Contribution Prerequisites: Design or Product Specification

Template & source document: No template. domain tool outputs (Gerber, STEP, e.g. mechanical or electrical design), spreadsheets (BOM ...), etc.

Allowable supporting documents allowed: None

ToDo: align to and update this document [doc](#).

ToDo: update section when Marketplace definition is updated.

Source Code: software, firmware, and hardware-description files

Source Code refers to any human-readable set of instructions, descriptions, or hardware-definition artifacts that accompany a contribution. This includes, but is not limited to:

- Software (e.g., application code, scripts, utilities, libraries, schema examples)
- Firmware (e.g., device firmware, embedded controller code)
- Hardware-related code such as:
 - RTL files (e.g., Verilog, SystemVerilog)
 - Hardware description languages (e.g., VHDL)
 - Any other code artifacts required to build, operate, or validate the design

Source Code must accompany a specification and is optional, except when seeking OCP Accepted™, in which case it is mandatory.

Contribution Prerequisites: Design or Product Specification

Template & source document: No template. GitHub, Any human readable form

Allowable supporting documents allowed: None (contribution is contained within GitHub)

Tools: modeling, Construction, and analysis utilities

Tools are analytical, computational, or validation** utilities used to evaluate, construct, model, verify/validate or optimize aspects of a data-center (DC) or compute environment. They support evaluation of cost, thermal performance, system flow characteristics, component interoperability, and other key parameters necessary for designing, validating, and deploying modern compute solutions. All tools include a user guide that describes the purpose, inputs, workflow, assumptions, and output interpretation.

**Verification and validation work together to ensure a product is built correctly with each focusing on different aspects of quality. This tools description simplifies by using only the term “validation” and only discussing in the design and implementation stages. Verification and validation contributions can apply across development, deployment, and operational phases.

Table 1: Tools Examples

Category	Description	Example Tools
Economic & Financial Modeling	Tools that model CAPEX/OPEX, budget impact, and financial returns.	TCO Calculators; ROI Models; Power & Utility Cost Estimators
Performance & Workload Characterization	Tools that measure, benchmark, or profile workload behavior and system performance.	Benchmarks (CPU/GPU/memory/storage); Workload Profilers; Synthetic Load Generators; Application Readiness Tools
Configuration & System Assessment	Tools that validate component choices, compatibility, and system correctness.	Configuration Assessment Tools; Interoperability/Compliance Checkers; BOM Generators; Power & Rail Validation Tools
Thermal, Mechanical & Environmental Modeling	Tools that analyze thermals, airflow, fluid behavior, and environmental constraints.	Thermal Simulation Tools; Airflow/CFD Models; Liquid Cooling Analysis Tools; Rack Density/Placement Tools
Power & Infrastructure Planning	Tools for evaluating electrical requirements, resilience, and energy efficiency.	Power Capacity Planners; Energy Efficiency Assessment Tools; UPS/Battery Runtime Calculators
Deployment, Validation & Troubleshooting	Tools that validate system readiness and diagnose issues during deployment or operation.	Deployment Validators; Diagnostic/Stress Tools; Telemetry Dashboards; Firmware/Health Scanners
Modeling & Design Automation	Tools for early-stage design, layout generation, and digital-twin simulation.	Rack Layout Planners; DC Capacity Planners; Digital Twins; Schematic/Blueprint Generators
Workflow Integration & Optimization	Tools that automate processes, map topologies, and optimize capacity or flow.	Automation Toolkits; Topology Mappers; Capacity Optimization Tools

Contribution Prerequisites: None

Template & source document: No template. Any format necessary to implement the tool.

This includes: Source code (GitHub), Spreadsheet, domain tool (e.g. mechanical or electrical design)

273 *Allowable supporting documents allowed:* None (expect all material within the submitted
274 package)

275

5 Validation & Operations

Two contribution types provide real-world results and operational capabilities:

- Tested Configurations
- Benchmarks

Tested Configuration: A validated combination of product(s), parameters, and deployment evidence.

A documented and tested configuration using an OCP Inspired™ or OCP Accepted™ Product and configured with additional hardware and operating system software and application software (specific use). A tested configuration includes testing plan/results as well as configuration details. The contributor is responsible for accuracy of information and warrants that the tested configuration will work in the defined use case and can provide support service.

Contribution Prerequisites: OCP Inspired™ or OCP Accepted™ Product

Template & source document: Document Template (OCP-aligned Markdown, .docx, Google Docs)

Allowable supporting documents allowed: Supporting data packages of various formats (spreadsheet, domain tool, etc.)

Benchmarks: A reference point or standard used for comparison

Benchmarks are structured evaluations used to compare business processes or performance metrics. Common metrics include cost, throughput, response time, and Power Usage Effectiveness (PUE). Benchmarking highlights real, collected data and can be used to compare traditional hardware, OCP-specification-compliant hardware, software performance, or any combination of these. Benchmarks should include clear data presentations, and while providing analysis is considered best practice, it is not strictly required. All benchmark contributions must reference the actual, relevant data and metrics collected to ensure transparency, accuracy, and reproducibility.

Contribution Prerequisites: None

Template & source document: Document Template (OCP-aligned Markdown, .docx, Google Docs)

Allowable supporting documents allowed: Supporting data packages of various formats (spreadsheet, domain tool, etc.)

ToDo: consider if there is any need for a test/val contribution (test requirements, checklists ...) that cannot be implemented as requirements document or as part of a specification.

6 Maintenance

Maintenance focuses on the ongoing support and evolution of a contribution through errata and maintenance releases.

Errata: Post-release defects, workarounds, and corrections

Errata are used to document corrections or clarifications to a contribution without modifying the original submitted material. Errata notify the Community of known defects, errors, omissions, instances of non-compliance, or identified issues along with any available workarounds or mitigations. Errata are permanently associated with the original submission, with the expectation that these corrections will be incorporated into a future version of the contribution.

Contribution Prerequisites: Original OCP Contribution

Template & source document: Errata Template (OCP-aligned Markdown, .docx, Google Docs) as well as same format as the original contribution

Allowable supporting documents allowed: None

Maintenance Release: Scheduled updates to maintain a contribution

A maintenance release is a scheduled update (generally software or firmware allowable for any contribution or supporting documents to maintain the functionality/utility) whose primary purpose is to fix issues and keep the contribution stable, not to introduce major new features. Generally, this lifecycle management release focuses on correcting defects, improving performance, addressing security vulnerabilities, and making small, low-risk enhancements required to keep the product reliable and supported.

Examples:

- Bug fixes
- Security patches
- Performance or stability improvements
- Minor enhancements or refinements (Small, non-breaking improvements that don't alter core functionality)

Contribution Prerequisites: Original OCP Contribution

Template & source document: Same format as the original contribution

Allowable supporting documents allowed: None

7 Training

Training is focused on educating the Community through the OCP Academy.

OCP Academy: Community Training

The OCP Academy online training courses leverage the vast OCP knowledge base, which includes papers, designs, specifications, case studies, webinars, and Summit recordings, to create a comprehensive online learning portal for data center technical content. OCP Academy helps grow Community participation and increase membership value.

Contribution Prerequisites: Any OCP contribution, webinar, or presentation

Template & source document: [online form](#)

8 Summary

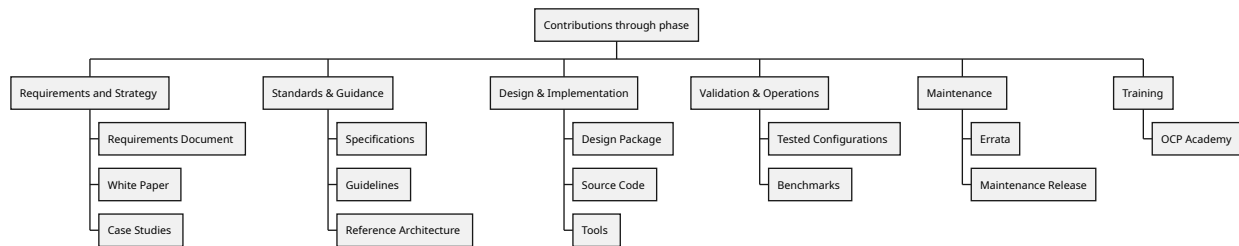


Figure 4: Mapping to contribution, alternate view

These usage categories help guide the Community in developing, classifying, and applying OCP contributions effectively and consistently.

Table 2: Usages

Usage	Purpose	Artifacts
Requirements & Strategy	Shape Direction	White Papers
		Requirements Document
		Case Studies
Standards & Guidance	Definition	Specifications
		Guidelines
		Reference Architecture
Design & Implementation	Design support	Design Packages
		Source Code
		Tools
Validation & Operations	Proof Points	Tested Configuration
		Benchmarks
Maintenance	Errata	Interim corrections
	Maintenance Release	Lifecycle Management
Training	OCP Academy	Community Education

Appendix A: Contribution Decision flow

This flow diagram helps to categorize the contribution based on the intent.

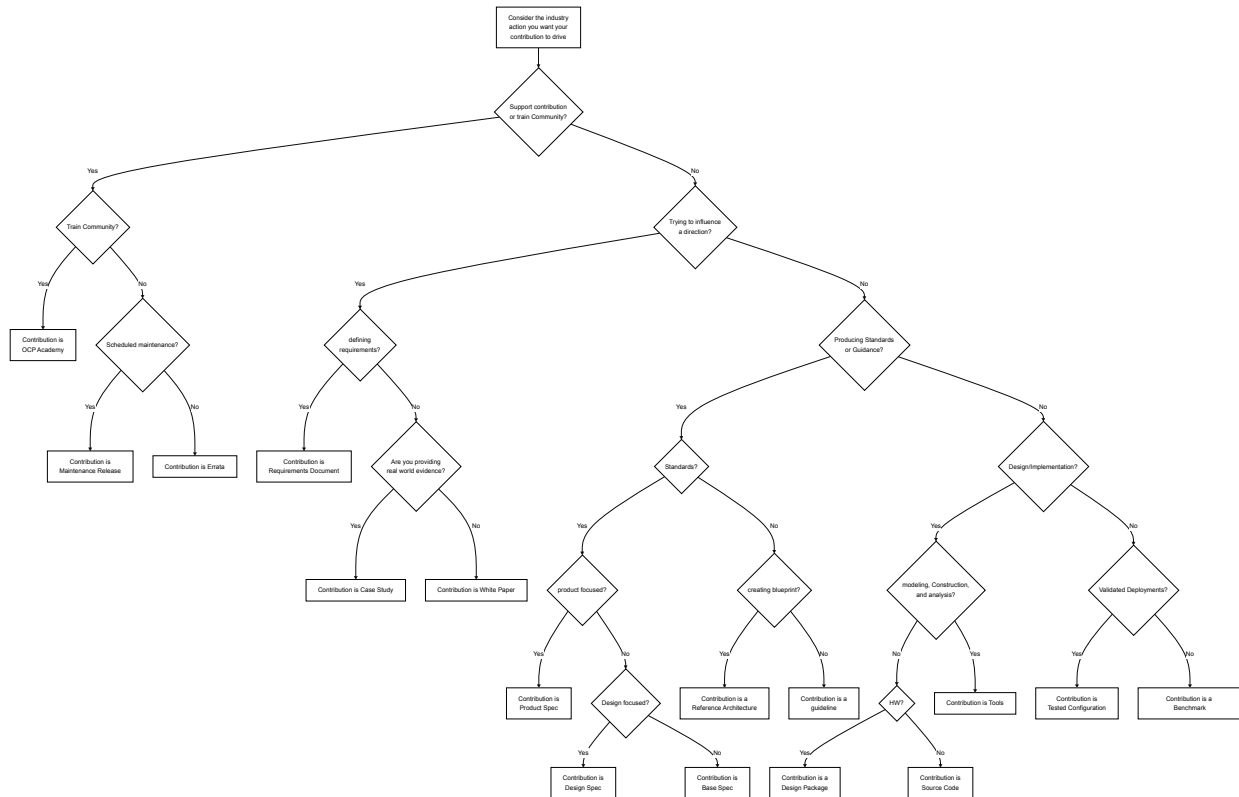


Figure 5: Choosing a Contribution

Appendix B: Contribution Evolution Examples

The contribution process may be straightforward, involving a single contribution, or more complex, spanning multiple contributions across development phases. The following examples illustrate potential contribution-stream scenarios throughout these phases.

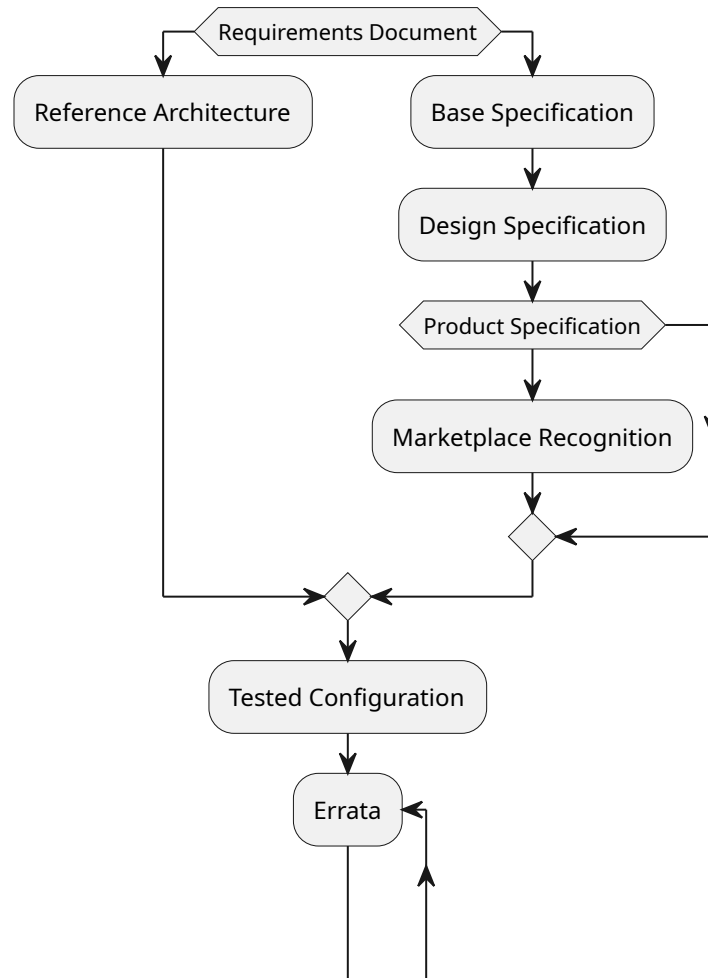


Figure 6: Example: Requirements Document to Specification

In this example, the contribution stream begins with a Requirements document that defines what the solution must achieve. Subsequent phases may include developing a Reference Architecture that satisfies the requirements, producing test results (Tested Configuration), and issuing updates through errata as needed. Alternatively, the Requirements document may lead to a full sequence of specifications (Base, Design, Product with options for a product recognition), which then flow into Tested Configurations and errata in the same manner as the other path.

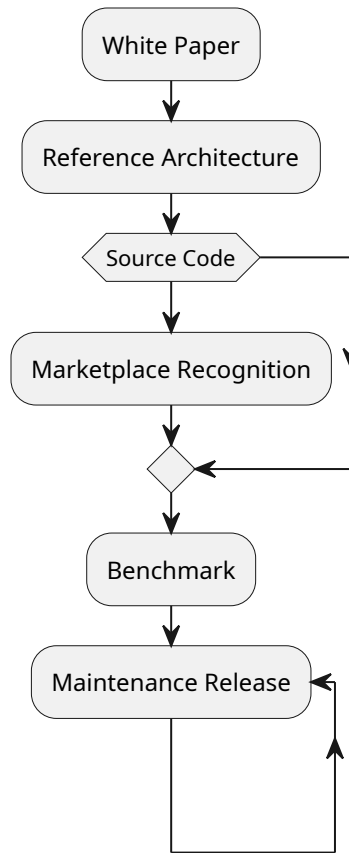


Figure 7: Example: White Paper to Source code

367 In this example, the contribution stream begins with a White Paper which provides
368 best practices on a particular issue (for this example let's say focused on Data Center
369 management). This is followed by a Reference Architecture which defines a solution
370 (blueprint) that guides the design and implementation of this DC management solution. An
371 option at this point is to contribute Source Code which satisfies the Reference Architecture;
372 This source code could be recognized in the OCP Marketplace™ if productized. This
373 could then be followed by a Benchmark contribution, and maintained through periodic
374 Maintenance Releases.