OR-001a — Developer Operations

Go to OR-001 — Operation by Identified User Categories

Statement

Crucible SHALL enable a Developer to perform each Crucible operation identified for the Developer by the applicable operational scenario.

Derived From

This requirement derives from:

- OR-001 — Operation by Identified User Categories

The Original Requirement states:

> *The system SHALL support operation by:*
>
> > *Developers*
> > *DevSecOps Engineers*
> > *Platform Engineers*
> > *System Administrators*
> > *Security Engineers*
> > *Compliance Officers*[C1]

OR-001a preserves the portion of the Original Requirement that requires Crucible to support operation by Developers.

The separate requirements derived from OR-001 address:

- OR-001b — DevSecOps Engineer Operations
- OR-001c — Platform Engineer Operations
- OR-001d — System Administrator Operations
- OR-001e — Security Engineer Operations
- OR-001f — Compliance Officer Operations

Assessment

The Original Requirement uses **operation** without identifying the actions performed by a Developer.

The term **operation** can refer to:

- Initiating a Crucible workflow
- Providing or selecting Controlled Inputs

- Initiating a Machine Image build
- Monitoring a build
- Reviewing build results
- Reviewing captured dependencies
- Initiating an Infrastructure Deployment
- Monitoring an Infrastructure Deployment
- Reviewing generated Evidence
- Investigating a failed operation

The Original Requirement does not identify:

- The Crucible operations performed by a Developer
- The operational scenarios in which the Developer participates
- Whether the Developer initiates, executes, configures, reviews, monitors, or observes an operation
- The interfaces used by the Developer
- The inputs provided by the Developer
- The outputs received or reviewed by the Developer
- The conditions applicable to the Developer operation
- The acceptance condition demonstrating operation by a Developer

OR-001a:

- Identifies Crucible as the responsible system
- Replaces **support** with the observable action **enable**
- Identifies a Developer as the applicable user category
- Identifies the applicable operational scenario as the source of the Developer operations
- Avoids introducing an Operational Role, role-assignment mechanism, or role-based authorization model
- Separates Developer operation from operation by the other user categories listed in OR-001

The Statement uses **each Crucible operation identified for the Developer** to apply one consistent enablement obligation across the operations assigned to the Developer by the applicable operational scenario.

When an operational scenario identifies independently testable Developer activities with different actors, objects, conditions, or completion criteria, those activities should be expressed through existing atomic requirements or additional requirements rather than embedded within OR-001a.

Concept of Operations Interpretation

The Concept of Operations identifies the following high-level Crucible operations:

- **Build** — Crucible builds Machine Images while applying the applicable image-hardening configuration
- **Capture** — Crucible captures build dependencies and compliance findings into managed storage
- **Deploy** — Crucible provisions infrastructure and executes the applicable pre-deployment and post-deployment steps

The cited Concept of Operations passage does not, by itself, state which of these operations a Developer performs.

A Developer can participate in a Crucible operation by:

- Initiating the operation
- Providing inputs to the operation
- Selecting applicable configuration
- Monitoring execution
- Reviewing results
- Responding to failures

The applicable operational scenario should identify the Developer participation relevant to each operation.

OR-001a does not independently allocate Build, Capture, or Deploy to the Developer. The allocation must remain traceable to the Concept of Operations, an architecture section, or an existing requirement.

Requirement Reuse Assessment

Before creating a lower-level requirement to realize OR-001a, the requirements corpus should be checked for an existing requirement that already defines the applicable Developer behavior.

An existing requirement can be reused when:

- The required Crucible behavior is the same
- The subject and object of the behavior are compatible
- The requirement applies to Developer participation
- The operational conditions cover the applicable scenario
- The verification outcome demonstrates Developer operation
- Reuse does not broaden or change the meaning of the existing requirement

The following requirements should be examined for reuse:

- Image-management requirements governing Machine Image Build
- Configuration-management requirements governing Developer-provided configuration
- Dependency-capture requirements governing Capture
- Deployment-orchestration requirements governing Deploy
- DevSecOps integration requirements governing workflow initiation and monitoring
- Interface requirements governing graphical, command-line, application programming, and automation interfaces
- Security requirements governing authentication and authorization
- Logging requirements governing operational event records
- Evidence, Provenance, Auditability, and Traceability requirements governing preservation of operational results

A new requirement should be created only when no existing requirement defines the required Developer behavior with the correct actor, object, conditions, and acceptance criteria.

OR-001a establishes that Developers can perform their identified Crucible operations. It does not

duplicate the detailed requirements governing Build, Capture, Deploy, security, logging, or interfaces.

Rationale

Developers need access to the Crucible operations required to prepare, initiate, monitor, and evaluate development-related workflows.

Developer participation can include:

- Preparing source content
- Preparing configuration content
- Providing Controlled Inputs
- Selecting an applicable build or deployment description
- Initiating a permitted workflow
- Monitoring workflow execution
- Reviewing generated Artifacts
- Reviewing workflow results
- Reviewing generated Evidence
- Investigating errors and failed operations

The applicable operational scenario determines which activities apply to the Developer.

Enabling Developer operations supports:

- Development workflow execution
- Early validation of configuration changes
- Repeatable Machine Image Build
- Review of captured dependencies
- Evaluation of deployment results
- Identification of defects before promotion
- Integration with DevSecOps workflows

This requirement does not require Developers to perform every Crucible operation.

This requirement does not establish the authorization model governing Developer access. Applicable Security Requirements establish authentication, authorization, and access-control behavior.

This requirement does not define the detailed Build, Capture, or Deploy behavior. Existing functional requirements should define and realize those operations.

Applies To

This requirement applies to:

- Crucible
- Developers
- Operational scenarios
- Developer activities

- Build operations
- Capture operations
- Deploy operations
- Controlled Inputs
- Machine Images
- Infrastructure Baselines
- Infrastructure Deployments
- User interfaces
- Command-line interfaces
- Application programming interfaces
- Automation interfaces
- Workflow results
- Failure and exception results
- Evidence
- Provenance
- Traceability

Verification

Verification confirms that:

1. The applicable operational scenario is identified
2. The Developer operations identified by the operational scenario are enumerated
3. The interface applicable to each Developer operation is identified
4. The inputs required by each Developer operation are identified
5. The outputs produced or presented by each Developer operation are identified
6. A Developer can perform each identified Developer operation
7. Each operation produces an identified result
8. Failed operations produce an observable result
9. Existing requirements used to realize each Developer operation are identified
10. The verification record preserves the required Evidence, Provenance, and Traceability

Verification includes:

- Operational scenario inspection
- Developer activity inspection
- Requirement reuse inspection
- Interface inspection
- Controlled Input inspection
- Developer workflow testing
- Build-operation testing, when applicable
- Capture-operation testing, when applicable
- Deploy-operation testing, when applicable
- Workflow monitoring testing
- Result-review testing
- Failure and exception testing
- Evidence inspection
- Provenance inspection
- Traceability inspection

The verification record identifies:

1. The applicable operational scenario
2. The Developer
3. Each identified Developer operation
4. The interface used for each operation
5. The inputs provided for each operation
6. The existing requirement or requirements realizing each operation
7. The result of each operation
8. Each generated Artifact
9. Each generated failure or exception result
10. Each identified coverage gap
11. The observed result
12. The generated Evidence

Requirements Realized By

This requirement is realized by applicable existing requirements governing:

- Configuration management
- Machine Image management
- Deployment orchestration
- DevSecOps integration
- Dependency capture
- User and automation interfaces
- Authentication and authorization
- Logging
- Evidence, Provenance, Auditability, and Traceability

Specific realization links should be added only after confirming that each referenced requirement provides Developer access to the applicable operation.

Related Architecture Sections

- 5. Descriptions, Composition, and Baselines
- 6. Machine Images and Image Layers
- 7. Infrastructure and Deployment
- 8. Dependencies and Air Gap Operations
- 10. Reproducibility, Provenance, and Traceability

Referenced By

The following pages reference this requirement:

- OR-001b — DevSecOps Engineer Operations

- OR-001c — Platform Engineer Operations
- OR-001d — System Administrator Operations
- OR-001e — Security Engineer Operations
- OR-001f — Compliance Officer Operations
- Annex D: Requirements Traceability

Delivery Phase

Phase 1 and subsequent phases

Implementation Status

Not Assessed

Implementation status requires identification of the Crucible operations assigned to Developers by the applicable operational scenarios and verification that Developers can perform those operations.

Requirement Status

Draft

This requirement derives from OR-001 in the Crucible System Requirements Specification, Version 1.1 Draft.

Notes for Editors

This requirement page should retain the stable requirement identifier `OR-001a`.

This page is a leaf requirement page and omits a trailing `:start` from its namespace.

Changes to the Statement should preserve the requirement that Developers can perform the Crucible operations identified for them by applicable operational scenarios.

Do not introduce an Operational Role, role-assignment mechanism, authorization model, or access-control model unless a controlling source establishes that concept.

Before creating an additional Developer-operation requirement:

- Review the applicable Concept of Operations scenario
- Identify the specific Developer activity
- Search the existing requirements corpus for equivalent behavior
- Reuse an existing requirement when its actor, behavior, object, conditions, and verification outcome provide the required coverage

- Create a new requirement only when the reuse review identifies a coverage gap

The operational scenario should identify:

- The Developer activity
- The applicable Crucible operation
- The interface used
- The inputs provided
- The outputs received or reviewed
- The applicable starting condition
- The applicable completion condition
- The applicable failure condition
- The existing requirements governing the operation
- The required Evidence

Material changes should receive review and should update the related verification criteria, requirements realization, related architecture sections, and source records.

To reference this requirement Statement from another wiki page, insert:

```
{{section>dido:02-crusible:99-annexes:annex-c-requirements:02-operational-requirements:or-001:or-001a#Statement&noheader&nofooter&noeditbtn}}
```

Do not rename this page after an external citation unless a redirect or move plan is in place.
