5.3 Capture and Preserve Dependencies

Go to 5. Operational Concept

Dependency Capture identifies, obtains, records, and preserves the Dependencies required to perform a selected Operational Lifecycle activity.

Crucible treats Dependency Capture as a controlled lifecycle activity rather than an informal packaging step. The captured dependency set provides the inputs needed to reproduce a construction, assessment, transfer, deployment, validation, maintenance, or recovery activity without relying on undocumented or unavailable resources.

Dependency Identification

Crucible identifies the direct and transitive Dependencies required by the selected lifecycle activity.

Dependencies can include:

- Source files
- Software packages
- Package metadata
- Build tools
- Compiler and interpreter versions
- Generated source and generated artifacts
- Base images
- Container build inputs
- Image Layers
- Infrastructure Baselines
- Configuration files
- Compliance content
- Provider tools and plugins
- Scripts
- Runtime libraries
- Environment assumptions
- Other inputs required to reproduce or operate the result

A direct Dependency is explicitly referenced by the selected input or lifecycle activity. A transitive Dependency is required by another Dependency.

Dependency identification must account for both forms when the selected activity cannot complete without them.

Dependency Capture

Crucible captures an identified Dependency from an authorized source.

The capture activity records information needed to identify and retrieve the Dependency, including:

- Dependency identifier
- Dependency type
- Version or revision
- Source or origin
- Retrieval location
- Capture time
- Integrity information
- Applicable license or usage information, when available
- The lifecycle activity requiring the Dependency
- The relationship to other Dependencies

Crucible does not treat an unspecified latest version, mutable tag, undocumented workstation file, or unrecorded external service as a reproducible Dependency.

When an external resource cannot be captured directly, the resulting record identifies the unresolved Dependency and the effect on the selected lifecycle activity.

Dependency Manifest

A Dependency Manifest identifies the Dependencies associated with a controlled lifecycle activity or resulting artifact.

The Dependency Manifest records:

- Each identified Dependency
- The selected version or revision
- The Dependency source
- The relationship between direct and transitive Dependencies
- The expected integrity value
- The capture status
- The preservation status
- Any unresolved or unavailable Dependency
- The artifact, image, Baseline, deployment, or activity that uses the Dependency

The Dependency Manifest provides a controlled description of the dependency set. The manifest does not replace the preserved Dependency content.

Dependency Records

A Dependency Record preserves information about an individual Dependency and its use.

A Dependency Record can identify:

- The Dependency identifier
- The selected revision
- The origin

- The integrity value
- The capture operation
- The preservation location
- The consuming artifact or lifecycle activity
- The applicable Dependency Manifest
- The verification result
- The associated Provenance
- The associated Traceability

Dependency Records allow a reader or automated process to determine which Dependency contributed to a result and where the preserved content resides.

Dependency Preservation

Crucible preserves captured Dependencies in a Dependency Store or another authorized repository.

Preservation protects the Dependency from loss, unrecorded replacement, or reliance on a source that may later become unavailable.

The preservation activity records:

- The preserved Dependency
- The preservation location
- The preserved revision
- The integrity value
- The preservation result
- The applicable retention information
- The relationship to the Dependency Manifest and Dependency Record

Preserving a Dependency does not authorize its use. Applicable security, licensing, compliance, export, import, and organizational controls continue to govern the Dependency.

Dependency Verification

Crucible verifies that a preserved Dependency corresponds to the identified Dependency.

Verification can include:

- Comparison of content digests
- Verification of a Digital Signature
- Confirmation of the Dependency identifier and revision
- Confirmation of the source or origin
- Confirmation that the preserved content remains readable and retrievable
- Confirmation that the Dependency Manifest and Dependency Record identify the preserved content correctly

A successful integrity check establishes that the evaluated content matches the recorded value. It does not independently establish that the Dependency is secure, compliant, licensed, approved, or

suitable for a particular use.

Connected and Disconnected Use

In a Connected Environment, Crucible can capture Dependencies from authorized external repositories and services.

Before operating in a Disconnected Environment or Air-Gapped Environment, Crucible must identify and preserve the Dependencies required by the selected lifecycle activity.

The preserved dependency set can be:

- Included in a Transfer Bundle
- Transferred through an authorized process
- Imported into the destination environment
- Stored in a local Dependency Store
- Used to populate Offline Repositories
- Verified before disconnected execution

The disconnected lifecycle activity must not depend on an unauthorized or unavailable resource outside the applicable operational boundary.

Capture and Preservation Result

The capture and preservation result identifies:

- The selected lifecycle activity
- The Dependency Manifest
- The Dependency Records
- The identified direct and transitive Dependencies
- The capture status of each Dependency
- The preservation status of each Dependency
- The preservation locations
- The integrity and verification results
- Any missing, unresolved, or unauthorized Dependency
- The associated Provenance
- The associated Traceability
- The generated Evidence

The preserved dependency set becomes a controlled input to subsequent construction, assessment, transfer, deployment, validation, maintenance, reproduction, or recovery activities.

Requirements Addressed

Requirement	Statement
FR-DEPC-001 — Capture Build Dependencies	Crucible SHALL perform Dependency Capture for Build Dependencies.
FR-DEPC-002 — Preserve Dependencies in a Dependency Store	Crucible SHALL preserve captured Build Dependencies in a managed Dependency Store.
FR-DEPC-003 — Produce a Transfer Bundle	Crucible SHALL produce a Transfer Bundle containing captured Build Dependencies.
FR-DEPC-004 — Populate Offline Repositories	Crucible SHALL populate Offline Repositories in a Disconnected Environment from a Transfer Bundle.
FR-DEPC-005 — Dependency Store Implementation Independence	Crucible SHALL permit the implementation of a Dependency Store to change without changing the Dependency Capture Contract.

The linked leaf requirement pages remain the canonical sources.
