STUNIR Glossary

A comprehensive glossary of terms used in STUNIR and related technologies.

A

Artifact

A file produced during the build process. Examples include IR files, generated code, and manifests.

Assembly

Low-level human-readable representation of machine code. STUNIR supports x86 and ARM assembly targets.

ASM

See Assembly.

Attestation

A signed statement that certifies properties of a build or artifact. Related to Provenance.

B

Build Determinism

See Determinism.

Build Receipt

See Receipt.

C

Canonical JSON

A standardized JSON format defined by RFC 8785 that ensures:
- Alphabetically sorted object keys
- No unnecessary whitespace
- Consistent number formatting
- Unicode normalization

This guarantees identical output regardless of implementation.

Canonicalization

The process of converting data to a standard, normalized form. Essential for Determinism.

CBOR

Concise Binary Object Representation. A binary data format similar to JSON. See also dCBOR.

Checksum

A value computed from data to detect errors or verify integrity. STUNIR uses SHA-256 for checksums.

D

dCBOR

Deterministic CBOR. A subset of CBOR that ensures deterministic encoding. Used for IR artifacts.

Determinism

The property that identical inputs always produce identical outputs. A core principle of STUNIR.

Example: Running `emit_ir.py` on the same spec file will always produce byte-for-byte identical IR.

Digest

See Hash.

E

Emitter

A component that generates target code from IR. STUNIR includes emitters for multiple languages.

Types:
- Built-in: Python, Rust, C89/C99, x86, ARM
- Custom: User-defined emitters

Epoch

A Unix timestamp (seconds since 1970-01-01). Used in IR and receipts to track generation time.

Export

A function declared as part of a module’s public API.

F

Fragment (Changelog)

A small file in `changelog.d/` describing a single change. Fragments are combined during release.

Function Signature

The definition of a function including its name, parameters, and return type.

G

Generator

See Emitter.

Git-cliff

A tool for generating changelogs from Git commit history. Used by STUNIR for release notes.

H

Hash

A fixed-size value computed from data using a cryptographic function. Used for integrity verification.

STUNIR uses SHA-256 for all hashes.

Hash Mismatch

An error indicating that computed hash differs from expected. Usually means file was modified.

I

Intermediate Representation (IR)

The canonical, normalized form of a specification. Properties:
- Language-independent
- Deterministic structure
- Includes computed metadata (hashes, epochs)

Structure:

```
{
  "ir_version": "1.0.0",
  "ir_epoch": 1738000000,
  "ir_spec_hash": "abc123...",
  "module": {...},
  "functions": [...]
}
```

IR Bundle

A collection of IR files for related modules.

J

JCS

JSON Canonicalization Scheme (RFC 8785). The standard STUNIR uses for Canonical JSON.

JSON

JavaScript Object Notation. The primary format for specs, IR, and manifests.

K

Key Ordering

The order of keys in a JSON object. Canonical JSON requires alphabetical ordering.

L

Legacy Mode

Compatibility mode for older STUNIR versions or formats.

M

Manifest

A file listing artifacts with their metadata (paths, hashes, sizes). Used for verification.

Types:
- **IR Manifest**: Lists IR files in `asm/ir/`
- **Targets Manifest**: Lists generated code files
- **Receipts Manifest**: Lists receipt files
- **Pipeline Manifest**: Describes build stages

Merkle Tree

A tree structure where each node contains a hash of its children. Future manifests may use this.

Module

A unit of code defined by a spec. Contains functions and metadata.

N

Native Tools

The Haskell-based toolchain for high-performance operations. Located in `tools/native/haskell/`.

O

Output

Generated artifacts from the STUNIR pipeline.

P

Parameter

An input to a function, defined by name and type.

Pipeline

The sequence of operations that transform a spec into verified output:
1. Spec Parse
2. IR Generation
3. Target Emission
4. Manifest Generation
5. Verification

Provenance

A record of how an artifact was created, including:
- Build timestamp
- Input hashes
- Tool versions
- Environment information

Used for auditing and reproducibility.

Q

Query (Manifest)

Searching or filtering manifest entries based on criteria.

R

Receipt

A cryptographic proof of a build operation. Contains:
- IR hash
- Spec hash
- Module information
- Self-referential receipt hash

Used for verification without rebuilding.

Reproducibility

The ability to recreate identical outputs from the same inputs. Related to Determinism.

RFC 8785

The specification for JSON Canonicalization Scheme. See Canonical JSON.

S

Schema

A definition of expected structure and types. Used in manifests and specs.

Semantic Versioning (SemVer)

Version numbering format: MAJOR.MINOR.PATCH
- MAJOR: Breaking changes
- MINOR: New features (backward compatible)
- PATCH: Bug fixes

SHA-256

Secure Hash Algorithm producing 256-bit (64 hex character) hashes. Used throughout STUNIR.

Signature

See Function Signature.

Spec (Specification)

The input to STUNIR defining a module's structure.

Required fields:
- `name` : Module name
- `version` : Semantic version
- `functions` : Function definitions
- `exports` : Public API

STUNIR

Spec **T**o **UN**iversal **I**ntermediate **R**epresentation.

A toolkit for generating deterministic, verifiable code from specifications.

Strict Mode

Verification mode that requires exact manifest matching with no extra files.

T

Target

The output language or platform for code generation.

Built-in targets:
- `python` : Python 3.x
- `rust` : Rust
- `c89` : ANSI C (C89)
- `c99` : C99
- `x86` : x86/x86_64 assembly
- `arm` : ARM/ARM64 assembly

Timestamp

See Epoch.

Type

A data type in the STUNIR type system.

Type	Description
`i32`	32-bit signed integer
`i64`	64-bit signed integer
`f32`	32-bit floating point
`f64`	64-bit floating point
`bool`	Boolean (true/false)
`str`	String
`void`	No value (for returns)

U

UN (Universal)

Indicates language/platform independence. The IR is "universal" because it can target any language.

V

Validation

Checking that data conforms to expected schema and constraints.

Verification

Confirming artifact integrity using manifests and hashes.

Version

A semantic version number (e.g., "1.0.0"). See Semantic Versioning.

W

Workflow

A defined sequence of operations. STUNIR uses a 5-phase workflow:
1. Spec Parse

2. IR Generation
3. Target Emission
4. Manifest Generation
5. Verification

X

x86

Intel/AMD processor architecture. STUNIR supports x86 and x86_64 assembly.

Y

YAML

YAML Ain't Markup Language. Alternative to JSON for configuration (not used for core STUNIR files).

Z

Zero-Knowledge

Cryptographic technique for proving knowledge without revealing it. Future consideration for STUNIR.

Acronyms

Acronym	Full Form
API	Application Programming Interface
ARM	Advanced RISC Machine
ASM	Assembly
CBOR	Concise Binary Object Representation
CI/CD	Continuous Integration/Continuous Deployment
CLI	Command Line Interface
dCBOR	Deterministic CBOR
IR	Intermediate Representation
JCS	JSON Canonicalization Scheme
JSON	JavaScript Object Notation
RFC	Request for Comments
SHA	Secure Hash Algorithm
STUNIR	Spec To Universal Intermediate Representation
WASM	WebAssembly

See Also

- User Guide (USER_GUIDE.md)
- API Reference (API.md)
- FAQ (FAQ.md)
- Tutorials (tutorials/)