STUNIR Module Organization

This document describes the module structure and organization of the STUNIR codebase.

Directory Structure

```
stunir_repo/
├── tools/                    # Core tooling
│   ├── cli.py                # Main CLI entry point
│   ├── ir_emitter/           # IR emission tools
│   │   ├── __init__.py
│   │   └── emit_ir.py        # Spec → IR conversion
│   ├── ir_canonicalizer/     # IR canonicalization
│   │   └── stunir-ir.cabal   # Haskell canonicalizer
│   ├── emitters/             # Code emitters
│   │   └── emit_code.py      # Target dispatch
│   ├── native/               # Native implementations
│   │   ├── haskell/          # Haskell native tools
│   │   │   └── stunir-native/
│   │   │       ├── src/
│   │   │       │   ├── Main.hs
│   │   │       │   └── Stunir/
│   │   │       │       ├── Manifest.hs
│   │   │       │       ├── Provenance.hs
│   │   │       │       ├── Canonical.hs
│   │   │       │       └── Receipt.hs
│   │   │       └── stunir-native.cabal
│   │   └── rust/             # Rust native tools
│   │       └── stunir-native/
│   │           ├── src/
│   │           │   ├── main.rs
│   │           │   ├── lib.rs
│   │           │   ├── canonical.rs
│   │           │   └── crypto.rs
│   │           └── Cargo.toml
│   └── conformance/          # Conformance testing
├── manifests/                # Manifest generation
│   ├── __init__.py
│   ├── base.py               # Shared utilities
│   ├── ir/                   # IR manifests
│   │   ├── gen_ir_manifest.py
│   │   └── verify_ir_manifest.py
│   ├── receipts/             # Receipt manifests
│   ├── contracts/            # Contract manifests
│   ├── targets/              # Target manifests
│   └── pipeline/             # Pipeline manifests
├── targets/                  # Target code generation
│   ├── asm/                  # Assembly targets
│   │   ├── ir/
│   │   └── README.md
│   ├── polyglot/             # High-level language targets
│   │   ├── c_base.py         # Shared C utilities
│   │   ├── rust/
│   │   ├── c89/
│   │   └── c99/
│   └── assembly/             # Low-level assembly
│       ├── base.py           # Shared assembly utilities
│       ├── x86/
│       └── arm/
├── spec/                     # Specification files
│   ├── targets/
│   │   └── polyglot/
│   │       └── skeletons/    # Project skeletons
│   └── schemas/              # JSON schemas
├── scripts/                  # Build and utility scripts
│   ├── build.sh              # Polyglot build entry
│   ├── verify.sh             # Verification script
│   ├── verify_strict.sh      # Strict verification
│   └── lib/                 # Script libraries
│       └── emit_dcbor.sh    # dCBOR emission
├── tests/                   # Test suite
│   ├── unit/                # Unit tests
│   ├── integration/         # Integration tests
│   └── determinism/         # Determinism tests
├── docs/                    # Documentation
│   ├── development/         # Developer guides
│   └── api/                 # API documentation
└── .github/                 # GitHub configuration
    ├── workflows/           # CI/CD workflows
    └── dependabot.yml       # Dependency updates
```

Module Hierarchy

Core Modules

```
stunir (root)
├── tools              # Tooling namespace
│   ├── cli            # CLI interface
│   ├── ir_emitter     # IR emission
│   └── emitters       # Code generation
├── manifests          # Manifest system
│   ├── base           # Shared utilities
│   ├── ir             # IR manifests
│   ├── receipts       # Receipt manifests
│   └── targets        # Target manifests
└── targets            # Code targets
    ├── asm            # Assembly output
    ├── polyglot       # Multi-language
    └── assembly       # Low-level asm
```

Import Guidelines

Within STUNIR

```
# Absolute imports preferred
from manifests.base import canonical_json, compute_sha256
from tools.ir_emitter.emit_ir import spec_to_ir
from targets.polyglot.c_base import CEmitterBase

# Relative imports for same package
from .base import BaseManifestGenerator
from ..utils import helper_function
```

Public API

```
# For external use, import from package root
from manifests import canonical_json, BaseManifestGenerator
from tools import emit_ir
```

Circular Dependency Prevention

1. **Base modules have no internal dependencies**
   - `manifests/base.py` - standalone utilities
   - `targets/polyglot/c_base.py` - standalone C utilities

2. **Type hints use forward references**
   ```python
   from **future** import annotations
   from typing import TYPE_CHECKING

if TYPE_CHECKING:
from .generator import ManifestGenerator
```

1. **Lazy imports for optional features**
   `python`
   `   def use_optional_feature():`
   `       from optional_module import feature`
   `       return feature()`

Module Responsibilities

`manifests/base.py`

- `canonical_json()` - RFC 8785 JSON canonicalization
- `compute_sha256()` - SHA-256 hashing
- `compute_file_hash()` - File hashing
- `scan_directory()` - Directory scanning
- `BaseManifestGenerator` - Abstract generator class
- `BaseManifestVerifier` - Abstract verifier class

`manifests/ir/`

- `gen_ir_manifest.py` - Generate IR manifests
- `verify_ir_manifest.py` - Verify IR manifests

`tools/ir_emitter/`

- `emit_ir.py` - Convert spec to IR
- Deterministic JSON output
- SHA-256 computation

`targets/polyglot/`

- `c_base.py` - Shared C89/C99 utilities
- `rust/emitter.py` - Rust target emitter
- `c89/emitter.py` - C89 target emitter
- `c99/emitter.py` - C99 target emitter

`targets/assembly/`

- `base.py` - Shared assembly utilities
- `x86/emitter.py` - x86 assembly emitter
- `arm/emitter.py` - ARM assembly emitter

Naming Conventions

Files

Type	Convention	Example
Module	snake_case	`emit_ir.py`
Generator	`gen_<type>_manifest.py`	`gen_ir_manifest.py`
Verifier	`verify_<type>_manifest.py`	`verify_ir_manifest.py`
Emitter	`emitter.py`	`targets/rust/emitter.py`
Base/Utils	`base.py`	`manifests/base.py`

Classes

Type	Convention	Example
Generator	`<Type>ManifestGenerator`	`IrManifestGenerator`
Verifier	`<Type>ManifestVerifier`	`IrManifestVerifier`
Emitter	`<Target>Emitter`	`RustEmitter`
Data class	`<Entity>`	`ManifestEntry`

Functions

Type	Convention	Example
Public	snake_case	`compute_sha256()`
Private	_snake_case	`_emit_statement()`
Factory	create_	`create_manifest()`
Converter	to	`spec_to_ir()`

Public/Private Boundaries

Public Exports

Define in `__init__.py`:

```python
# manifests/__init__.py
from .base import (
    canonical_json,
    compute_sha256,
    compute_file_hash,
    scan_directory,
    BaseManifestGenerator,
    BaseManifestVerifier,
)

__all__ = [
    "canonical_json",
    "compute_sha256",
    "compute_file_hash",
    "scan_directory",
    "BaseManifestGenerator",
    "BaseManifestVerifier",
]
```

Private Functions

Prefix with underscore:

```python
def _internal_helper():
    """Internal helper, not part of public API."""
    pass

class MyClass:
    def _private_method(self):
        """Private method."""
        pass
```

Testing Organization

```
tests/
├── conftest.py           # Shared fixtures
├── unit/                 # Unit tests (isolated)
│   ├── test_canonical.py
│   ├── test_sha256.py
│   └── manifests/
│       ├── test_base.py
│       └── test_ir_manifest.py
├── integration/          # Integration tests
│   ├── test_full_pipeline.py
│   └── test_manifest_workflow.py
└── determinism/          # Determinism verification
    ├── test_json_determinism.py
    └── test_hash_determinism.py
```

Test Naming

```python
def test_compute_sha256_with_bytes():
    """Test SHA-256 with bytes input."""
    pass

def test_compute_sha256_with_string():
    """Test SHA-256 with string input."""
    pass

def test_compute_sha256_empty_raises_error():
    """Test SHA-256 raises ValueError for empty input."""
    pass
```