Week 8: Python IR Pipeline Investigation Report

Date: January 31, 2026
Investigator: AI Assistant
Status: ISSUES IDENTIFIED - FIXES IN PROGRESS

Executive Summary

The Python pipeline (`tools/spec_to_ir.py`) generates **incorrect IR format** that does not match the `stunir_ir_v1` schema used by Rust and SPARK implementations. The main issue is in the `steps` array structure within function definitions.

Investigation Findings

1. Schema Comparison

Top-Level Structure ✅ CORRECT

All three implementations (Rust, SPARK, Python) generate the correct top-level structure:

```
{
  "schema": "stunir_ir_v1",
  "ir_version": "v1",
  "module_name": "mavlink_handler",
  "docstring": "...",
  "types": [],
  "functions": [...]
}
```

Status: Python implementation is CORRECT at the top level.

---

2. Steps Format ❌ INCORRECT

Expected Format (per stunir_ir_v1.schema.json)

According to `/home/ubuntu/stunir_repo/schemas/stunir_ir_v1.schema.json` , steps should have:

```
{
  "op": "assign|return|call|error",  // Required
  "target": "variable_name",          // Optional
  "value": "expression"               // Optional (any type)
}
```

Rust Pipeline Output ✅ CORRECT

File: `/home/ubuntu/stunir_repo/test_outputs/rust_pipeline/ir.json`

```
{
  "op": "assign",
  "target": "msg_type",
  "value": "buffer[0]"
}
```

Status: CORRECT - Matches schema exactly.

SPARK Pipeline Output ✅ ACCEPTABLE

File: `/home/ubuntu/stunir_repo/test_outputs/spark_pipeline/ir.json`

```
{
  "op": "noop"
}
```

Status: SIMPLIFIED but valid - Uses minimal “noop” operations. Does not fully capture statement semantics but is schema-compliant.

Python Pipeline Output ❌ INCORRECT

File: `/home/ubuntu/stunir_repo/test_outputs/python_test/ir.json`

```
{
  "kind": "var_decl",  // ❌ Wrong field name - should be "op"
  "data": "{'type': 'var_decl', 'var_name': 'msg_type', 'var_type': 'u8', 'init':
'buffer[0]'}"  // ❌ Stringified dict - should be separate fields
}
```

Issues:
1. Uses `"kind"` instead of `"op"` field
2. Uses `"data"` field containing a stringified Python dict
3. Does not use structured `"target"` and `"value"` fields
4. The `"data"` field is not part of the stunir_ir_v1 schema

---

3. Type Conversion Differences

Type Mapping

Spec Type	Python Output	Rust Output	Expected
`byte[]`	`bytes`	`byte[]`	`byte[]` ✅
`u8`	`u8` ✅	`u8` ✅	`u8` ✅
`i32`	`i32` ✅	`i32` ✅	`i32` ✅

Issue: Python converts `byte[]` to `bytes`, while Rust keeps it as `byte[]`.

Root Cause Analysis

Location: `tools/spec_to_ir.py`, lines 143-158

```
# Convert body - preserve original structure with kind/data fields
steps = []
for stmt in func_spec.get("body", []):
    # If statement has 'type' field, convert to 'kind' and preserve in 'data'
    if "type" in stmt:
        step = {
            "kind": stmt.get("type", "nop"),  # ❌ Should be "op"
            "data": str(stmt)  # ❌ Should parse stmt into target/value
        }
    else:
        # Fallback for simpler format
        step = {"op": stmt.get("op", "nop")}  # ✅ This branch is correct
        if "target" in stmt:
            step["target"] = stmt["target"]
        if "value" in stmt:
            step["value"] = stmt["value"]
    steps.append(step)
```

Root Cause:
The first branch (lines 146-150) uses the wrong field names and stringifies the entire statement instead of extracting structured fields.

Required Fixes

Fix 1: Update `convert_spec_to_ir()` Function

File: `tools/spec_to_ir.py`, lines 143-158

Current Code (WRONG):

```
if "type" in stmt:
    step = {
        "kind": stmt.get("type", "nop"),
        "data": str(stmt)
    }
```

Fixed Code:

```
if "type" in stmt:
    # Map statement type to IR op
    stmt_type = stmt.get("type", "nop")
    op_map = {
        "var_decl": "assign",
        "assign": "assign",
        "return": "return",
        "call": "call",
        "comment": "nop",
        "if": "call",
        "loop": "call"
    }

    step = {"op": op_map.get(stmt_type, "nop")}

    # Extract target from var_name or target field
    if "var_name" in stmt:
        step["target"] = stmt["var_name"]
    elif "target" in stmt:
        step["target"] = stmt["target"]

    # Extract value from init, value, or other fields
    if "init" in stmt:
        step["value"] = stmt["init"]
    elif "value" in stmt:
        step["value"] = stmt["value"]
    elif "func_name" in stmt:
        step["value"] = stmt["func_name"]
```

Fix 2: Update Type Conversion

File: `tools/spec_to_ir.py`, lines 65-83

Current Code:

```
"byte[]": "bytes",  # ❌ Wrong
```

Fixed Code:

```
"byte[]": "byte[]",  # ✅ Keep original type
```

Testing Plan

Test 1: Generate IR with Python

```
cd /home/ubuntu/stunir_repo
python3 tools/spec_to_ir.py \
  --spec-root spec/ardupilot_test \
  --out test_outputs/python_pipeline/ir.json \
  --lockfile local_toolchain.lock.json
```

Test 2: Compare Outputs

```
# Compare Python vs Rust
jq '.functions[0].steps' test_outputs/python_pipeline/ir.json
jq '.functions[0].steps' test_outputs/rust_pipeline/ir.json

# Validate against schema
ajv validate -s schemas/stunir_ir_v1.schema.json -d test_outputs/python_pipeline/
ir.json
```

Test 3: Test ir_to_code

```
python3 tools/ir_to_code.py \
  --ir test_outputs/python_pipeline/ir.json \
  --out test_outputs/python_pipeline/output.c \
  --target c
```

Expected Outcomes After Fix

✅ Python generates `"op"` field instead of `"kind"`
✅ Python generates structured `"target"` and `"value"` fields
✅ Python output matches Rust IR structure
✅ Python IR validates against `stunir_ir_v1.schema.json`
✅ `ir_to_code.py` can parse Python-generated IR

Priority Ranking

1. **HIGH:** Fix step format (`kind` → `op` , structured fields)
2. **MEDIUM:** Fix type conversion (`bytes` → `byte[]`)
3. **LOW:** Add validation against schema in Python pipeline
4. **LOW:** Add tests for Python IR generation

Next Steps

1. ✅ Document investigation findings (this file)
2. 🔄 Apply fixes to `tools/spec_to_ir.py`
3. ⏳ Test Python IR generation
4. ⏳ Investigate `ir_to_code.py` compatibility
5. ⏳ Add missing templates
6. ⏳ Complete end-to-end Python pipeline test

References

- Schema: `/home/ubuntu/stunir_repo/schemas/stunir_ir_v1.schema.json`
- Python Tool: `/home/ubuntu/stunir_repo/tools/spec_to_ir.py`
- Rust Output: `/home/ubuntu/stunir_repo/test_outputs/rust_pipeline/ir.json`
- SPARK Output: `/home/ubuntu/stunir_repo/test_outputs/spark_pipeline/ir.json`
- Python Output: `/home/ubuntu/stunir_repo/test_outputs/python_test/ir.json`

Report Status: Complete
Next Action: Apply fixes to Python implementation