Phase 3.5: Interactive Config Builder - Test Plan

Prerequisites

- Package built and installed: `dbus-mqtt-bridge_0.1.0-1_amd64.deb`
- Terminal with ncurses support (any modern terminal)
- MQTT broker running (for testing generated configs)
- D-Bus system and/or session bus accessible

Test Suite 1: Basic Functionality

Test 1.1: Help Display for Generator

Objective: Verify --generate-config is documented in help

Steps:

```bash
dbus-mqtt-bridge --help
```

Expected Output:

- Help text includes `--generate-config` option
- Shows `--from FILE` option
- Shows `-o FILE` option
- Includes example: `dbus-mqtt-bridge --generate-config`

Verification:

- [ ] --generate-config option listed
- [ ] Examples shown
- [ ] Usage syntax clear

Test 1.2: Fresh Interactive Config Generation

Objective: Generate a new config from scratch

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. Enter MQTT broker: `localhost`
2. Accept default port: `1883` (press Enter)
3. Authentication: `n` (no)
4. Bus type: Select `system` (arrow keys + Enter)
5. Mappings: Select `[c] Continue` (skip for now)
6. Output path: `test-fresh.yaml`

Expected Behavior:

- ✓ Prompts appear in order
- ✓ Defaults shown in brackets
- ✓ Bus type uses ncurses selection
- ✓ Config saved successfully
- ✓ File contains valid YAML

Verification:

```bash
cat test-fresh.yaml
dbus-mqtt-bridge --validate-only test-fresh.yaml  # Should be valid
```

Checklist:

- [ ] All prompts displayed
- [ ] Navigation worked
- [ ] File created
- [ ] YAML structure correct
- [ ] Validation passes

Test 1.3: Output to Stdout

Objective: Config printed to stdout when no path given

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. Configure MQTT (any values)
2. Select bus type
3. Skip mappings
4. Output path: (press Enter for stdout)

Expected Output:

```
---
mqtt:
  broker: localhost
  port: 1883
...
---
```

Verification:

- [ ] YAML printed to stdout
- [ ] Formatted correctly
- [ ] Can redirect: `dbus-mqtt-bridge --generate-config > config.yaml`

Test Suite 2: D-Bus Introspection

Test 2.1: Service List Navigation

Objective: Browse D-Bus services with ncurses

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. MQTT config: defaults
2. Bus type: system
3. Add mapping: `[a] Add new mapping`
4. Service name: `list`

Expected Behavior:

- ✓ Ncurses UI appears
- ✓ Two sections: SYSTEM BUS and SESSION BUS
- ✓ Services listed with `[SYS]` and `[SES]` prefixes
- ✓ Can navigate with arrow keys
- ✓ Selection highlights current item
- ✓ Scroll indicators if list is long

Keyboard Tests:

- [ ] ↑ key moves up
- [ ] ↓ key moves down
- [ ] Page Up scrolls up
- [ ] Page Down scrolls down
- [ ] Home goes to first item
- [ ] End goes to last item
- [ ] Enter selects item
- [ ] 'q' cancels
- [ ] 'm' allows manual entry

Verification:

```bash
# Should see services like:
[SYS] org.freedesktop.systemd1
[SYS] org.freedesktop.NetworkManager
[SES] org.freedesktop.Notifications
```

Test 2.2: Service Selection and Bus Detection

Objective: Selecting service sets correct bus type

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. MQTT config: defaults
2. Bus type: system
3. Add D-Bus→MQTT mapping
4. Service: `list`
5. Select `[SES] org.freedesktop.Notifications` (session service)

Expected Behavior:

- ✓ Warning shown: "This is a SESSION bus service"
- ✓ Prompts to switch to session bus
- ✓ Shows implications of session bus

Verification:

- ☐ Bus mismatch warning displayed
- ☐ Prompted to switch
- ☐ If accepted, bus_type becomes "session"
- ☐ Implications shown (user service, ~/.config, etc.)

Test 2.3: Path Introspection

Objective: Discover object paths via introspection

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. Configure up to D-Bus service selection
2. Service: `org.freedesktop.NetworkManager` (or type manually)
3. Path: `introspect`

Expected Behavior:

- ✓ Shows "Introspecting org.freedesktop.NetworkManager..."
- ✓ Lists available paths
- ✓ Allows selection or manual entry

Verification:

- [ ] Introspection completes
- [ ] Paths listed (at least `/`)
- [ ] Can select from list
- [ ] Can still enter manually with 'm'

Test 2.4: Interface Discovery

Objective: List interfaces via introspection

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. Service: `org.freedesktop.NetworkManager`
2. Path: `/org/freedesktop/NetworkManager`
3. Interface: `introspect`

Expected Behavior:

- ✓ Introspects the path
- ✓ Lists available interfaces
- ✓ Includes standard interfaces (Introspectable, Properties, etc.)
- ✓ Allows selection

Verification:

- [ ] Multiple interfaces shown
- [ ] Includes `org.freedesktop.NetworkManager`
- [ ] Selection works
- [ ] Manual entry still available

Test 2.5: Signal Discovery

Objective: Find available signals in an interface

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. Service: `org.freedesktop.NetworkManager`
2. Path: `/org/freedesktop/NetworkManager`
3. Interface: `org.freedesktop.NetworkManager`
4. Signal: `introspect`

Expected Behavior:

- ✓ Shows signals like `StateChanged`, `PropertiesChanged`
- ✓ Allows selection
- ✓ Ncurses navigation works

Verification:

- [ ] Signals listed
- [ ] Can select
- [ ] Selected signal name is valid

Test 2.6: Method Discovery (MQTT→D-Bus)

Objective: Find available methods for MQTT→D-Bus mappings

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. Configure MQTT and bus
2. Skip D-Bus→MQTT mappings
3. Add MQTT→D-Bus mapping
4. Configure topic: `test/topic`
5. Service/path/interface: (as above)
6. Method: `introspect`

Expected Behavior:

- ✓ Lists available methods
- ✓ Allows selection

Verification:

- [ ] Methods listed
- [ ] Can select
- [ ] Method name valid

Test Suite 3: Config Loading and Editing

Test 3.1: Load Partial Config

Objective: Load existing config and fill missing fields

Steps:

```bash
# Create partial config
cat > partial-test.yaml <<'EOF'
mqtt:
  broker: mqtt.example.com
  port: 1883
EOF

dbus-mqtt-bridge --generate-config --from partial-test.yaml -o complete-test.yaml
```

Expected Behavior:

- ✓ Loads existing MQTT config
- ✓ Uses as defaults in prompts
- ✓ Skips prompts for valid fields
- ✓ Only asks for missing fields (bus_type, mappings)

Verification:

```bash
cat complete-test.yaml
# Should have mqtt.example.com as broker
# Should have bus_type added
```

Checklist:

- [ ] Existing values loaded
- [ ] Used as defaults
- [ ] Only prompted for missing fields
- [ ] Final config complete

Test 3.2: Fix Invalid Config

Objective: Load broken config and fix errors

Steps:

```bash
# Create invalid config
cat > broken-test.yaml <<'EOF'
mqtt:
  broker: "not a valid hostname!"
  port: 99999
bus_type: invalid_type
mappings:
  dbus_to_mqtt:
    - service: "bad-service-name"
      path: "no-leading-slash"
      interface: "org.example.Test"
      signal: "Signal"
      topic: "test/#"
  mqtt_to_dbus: []
EOF

dbus-mqtt-bridge --generate-config --from broken-test.yaml -o fixed-test.yaml
```

Expected Behavior:

- ✓ Loads config
- ✓ Validates immediately
- ✓ Shows all errors:
  - Invalid broker
  - Invalid port (> 65535)
  - Invalid bus_type
  - Invalid service name (no dots)
  - Invalid path (no leading /)
  - Invalid topic (wildcard in publish)
- ✓ Prompts to fix
- ✓ Re-prompts for each invalid field
- ✓ Final config is valid

Verification:

```bash
dbus-mqtt-bridge --validate-only fixed-test.yaml
# Should pass validation
```

Checklist:

- [ ] Errors detected
- [ ] All errors shown
- [ ] Prompted to fix
- [ ] Each error corrected
- [ ] Final validation passes

Test 3.3: Credentials Never Loaded

Objective: Verify passwords are never loaded from file (security)

Steps:

```bash
# Create config with credentials
cat > with-creds.yaml <<'EOF'
mqtt:
  broker: localhost
  port: 1883
  auth:
    username: testuser
    password: secretpassword
bus_type: system
mappings:
  dbus_to_mqtt: []
  mqtt_to_dbus: []
EOF

dbus-mqtt-bridge --generate-config --from with-creds.yaml
```

Expected Behavior:

- ✓ Loads other config
- ✓ ALWAYS prompts for authentication
- ✓ Does NOT show old password
- ✓ Requires re-entering credentials

Verification:

- [ ] Auth prompt appears
- [ ] No default password shown
- [ ] Must enter fresh credentials

Test Suite 4: Mapping Management

Test 4.1: Add D-Bus→MQTT Mapping

Objective: Create a D-Bus to MQTT mapping

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. MQTT: defaults
2. Bus: system
3. D-Bus→MQTT: `[a] Add new mapping`
4. Service: `org.freedesktop.NetworkManager` (or use `list`)
5. Path: `/org/freedesktop/NetworkManager`
6. Interface: `org.freedesktop.NetworkManager`
7. Signal: `StateChanged`
8. Topic: `network/state`
9. Continue: `[c] Continue`
10. Skip MQTT→D-Bus
11. Save config

Verification:

```bash
cat config.yaml
# Should have:
# mappings:
#   dbus_to_mqtt:
#     - service: org.freedesktop.NetworkManager
#       path: /org/freedesktop/NetworkManager
#       interface: org.freedesktop.NetworkManager
#       signal: StateChanged
#       topic: network/state
```

Checklist:

- [ ] Mapping added
- [ ] All fields correct
- [ ] Saved to config
- [ ] Config validates

Test 4.2: Edit Existing Mapping

Objective: Modify an existing mapping

Steps:

```bash
dbus-mqtt-bridge --generate-config --from config.yaml
```

Interactive Steps:

1. D-Bus→MQTT: `[e] Edit mapping`
2. Enter number: `1`
3. Change topic to: `system/network/state`
4. Keep other fields (press Enter for defaults)

Verification:

- [ ] Existing values shown as defaults
- [ ] Can modify individual fields
- [ ] Mapping updated in final config

Test 4.3: Delete Mapping

Objective: Remove a mapping

Steps:

```bash
dbus-mqtt-bridge --generate-config --from config.yaml
```

Interactive Steps:

1. D-Bus→MQTT: `[d] Delete mapping`
2. Enter number: `1`
3. Confirm: `y`

Verification:

- [ ] Confirmation prompt shown
- [ ] Mapping removed
- [ ] List updates

Test 4.4: Add MQTT→D-Bus Mapping

Objective: Create MQTT to D-Bus mapping

Steps:

```
bash

dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. Configure MQTT and bus
2. Skip D-Bus→MQTT
3. MQTT→D-Bus: `[a] Add new mapping`
4. Topic: `control/command`
5. Service/path/interface/method: (configure as needed)

Verification:

- [ ] Mapping added
- [ ] Topic field first (different order from D-Bus→MQTT)
- [ ] Method instead of signal

Test 4.5: Wildcard Topic Warning

Objective: Warn about wildcards in subscriptions

Steps:

```
bash

dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. Add MQTT→D-Bus mapping
2. Topic: `sensors/+/temperature/#`

Expected Behavior:

- ✓ Accepts topic (wildcards allowed in subscriptions)
- ✓ Shows warning about wildcard usage
- ✓ Validation passes with warning

Verification:

- [ ] Topic accepted
- [ ] Warning shown during validation
- [ ] Config still valid (warnings don't fail)

Test Suite 5: Error Handling

Test 5.1: Non-existent Service Warning

Objective: Warn when service doesn't exist

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. Service: `org.nonexistent.Service` (manual entry)

Expected Behavior:

- ✓ Warning: "Service not found on any bus"
- ✓ Prompts: "Continue anyway?"
- ✓ If yes, allows continuing
- ✓ If no, re-prompts

Verification:

- [ ] Warning shown
- [ ] User choice respected

Test 5.2: Introspection Failure

Objective: Handle services that don't support introspection

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. Service: (one that might not support introspection)
2. Path: `introspect`

Expected Behavior:

- ✓ Shows error message
- ✓ Falls back to manual entry
- ✓ Doesn't crash

Verification:

- [ ] Error handled gracefully
- [ ] Can continue with manual entry

Test 5.3: Invalid Permission on Output File

Objective: Handle write permission errors

Steps:

```bash
dbus-mqtt-bridge --generate-config
```

Interactive Steps:

1. Generate valid config
2. Output path: `/root/config.yaml` (no permission)

Expected Behavior:

- ✓ Error: "Could not write to /root/config.yaml"
- ✓ Prints: "Output would be:"
- ✓ Shows full config to stdout

Verification:

- [ ] Error message shown
- [ ] Config printed to stdout
- [ ] Can copy/paste output

Test Suite 6: Integration Testing

Test 6.1: Generated Config Works

Objective: Config generated by tool runs successfully

Steps:

```bash
# Generate config with real mapping
dbus-mqtt-bridge --generate-config -o /tmp/test-integration.yaml

# Use session bus for easier testing
# Add: org.freedesktop.Notifications → notifications/closed
```

Run the bridge:

```bash
dbus-mqtt-bridge /tmp/test-integration.yaml
```

Trigger signal:

```bash
notify-send --expire-time=2000 "Test" "Message"
```

Verification:

- [ ] Config loads without errors
- [ ] Bridge starts
- [ ] Validation passes at startup
- [ ] Signal triggered successfully
- [ ] (If MQTT monitoring) Message received

Test 6.2: Fixed Config Works

Objective: Broken config fixed by tool works

Steps:

```bash
# Create broken config (from Test 3.2)
# Fix it with generator
# Run the fixed config

dbus-mqtt-bridge fixed-test.yaml
```

Verification:

- [ ] Fixed config loads
- [ ] No validation errors
- [ ] Bridge runs

Test Summary Checklist

Core Functionality

- [ ] Interactive generation works
- [ ] Output to file works
- [ ] Output to stdout works
- [ ] Help shows --generate-config

D-Bus Introspection

- [ ] Service list appears
- [ ] Ncurses navigation works
- [ ] Bus detection works
- [ ] Path introspection works
- [ ] Interface discovery works
- [ ] Signal discovery works
- [ ] Method discovery works

Config Loading

- [ ] Partial config loads
- [ ] Invalid config detected
- [ ] Errors fixed interactively
- [ ] Credentials never loaded
- [ ] Existing values used as defaults

Mapping Management

- [ ] Add D-Bus→MQTT mapping
- [ ] Add MQTT→D-Bus mapping
- [ ] Edit mapping
- [ ] Delete mapping
- [ ] Wildcard warnings shown

Error Handling

- [ ] Non-existent service warnings
- [ ] Introspection failures handled
- [ ] Permission errors handled
- [ ] Invalid input rejected

Integration

- [ ] Generated config runs
- [ ] Fixed config runs
- [ ] Validation passes
- [ ] Actual bridging works

Success Criteria

All tests pass with:

- ✓ No crashes
- ✓ Clear error messages
- ✓ Intuitive navigation
- ✓ Valid output configs
- ✓ Working bridge functionality