Phase 3.5: Interactive Config Builder - Implementation Summary

Overview

Implemented a comprehensive interactive configuration builder with D-Bus introspection, ncurses-based navigation, and intelligent config editing/fixing capabilities.

Features Implemented

1. Interactive Config Generation (`--generate-config`)

Complete workflow:

```bash
# Start fresh config
dbus-mqtt-bridge --generate-config

# Edit/fix existing config
dbus-mqtt-bridge --generate-config --from config.yaml -o fixed-config.yaml

# Partial config completion
dbus-mqtt-bridge --generate-config --from partial.yaml
```

2. D-Bus Introspection

Discover services, signals, and methods:

- Lists services from both system and session buses
- Shows available interfaces on a service
- Displays signals and methods for interfaces
- Validates service availability on correct bus

Features:

- Combined system/session bus listing with clear labels
- Automatic bus type detection and warnings
- Error handling for unavailable services
- Graceful fallback when introspection fails

3. Ncurses Interactive Selection

Full keyboard navigation:

- ↑/↓ arrows: Navigate lists
- Page Up/Down: Fast navigation
- Home/End: Jump to start/end
- Enter: Select item
- 'q' / ESC: Cancel
- 'm': Manual entry

Features:

- Scrolling for long lists (D-Bus services can have 100+ entries)
- Visual indicators for more items above/below
- Highlighted selection
- Smooth navigation experience

4. Partial Config Support

Smart config loading:

```yaml
# partial-config.yaml - only has some fields
mqtt:
  port: 1883  # Valid
bus_type: system

# Generator will:
# 1. Load existing values
# 2. Use as defaults in prompts
# 3. Only ask for missing/invalid fields
# 4. Validate and fix errors
```

Security:

- Always prompts for `username` and `password` (never scripted)
- Credentials never loaded from file (security best practice)

5. Validation and Error Fixing

Automatic error detection:

- Validates config after generation
- Shows all errors with clear messages
- Offers to fix errors interactively
- Re-validates after fixes

Interactive fixing:

```
Configuration has errors:

  Field 'mqtt.broker': Invalid broker 'bad-hostname!'
  Field 'mappings.dbus_to_mqtt[0].service': Invalid service name

Would you like to fix these errors? [Y/n]:
```

6. Mapping Management

D-Bus → MQTT mappings:

- Add/Edit/Delete mappings
- List current mappings with indices
- Introspect to discover signals
- Validate each field

MQTT → D-Bus mappings:

- Add/Edit/Delete mappings
- Support for wildcards in subscriptions
- Introspect to discover methods
- Validate topics and method calls

7. Bus Type Implications

Clear guidance for users:

```
⚠ Bus Type Implications:
  • System bus selected
  • Requires root privileges or system service
  • Config should be in: /etc/dbus-mqtt-bridge/config.yaml
  • Requires D-Bus policy configuration
  • Run as: sudo systemctl enable --now dbus-mqtt-bridge
```

Shows different instructions for system vs session bus.

8. Output Handling

Flexible output:

- Prompt for output path at end
- Default to stdout if path invalid
- Show "would be" output on permission errors
- Provide next steps after saving

Files Created

New Headers

1. **`include/InteractiveSelector.h`**
   - Ncurses-based interactive selection
   - Text/password prompts
   - Yes/no prompts
2. **`include/DbusIntrospector.h`**
   - D-Bus service discovery
   - Introspection XML parsing
   - Signal/method extraction
3. **`include/ConfigGenerator.h`**
   - Main config generation interface
   - Mapping management
   - Validation and fixing

New Source Files

4. **`src/InteractiveSelector.cpp`**
    - Ncurses implementation (320 lines)
    - Keyboard navigation
    - Password masking
5. **`src/DbusIntrospector.cpp`**
    - sdbus-c++ based introspection (200 lines)
    - XML parsing with regex
    - Bus detection
6. **`src/ConfigGenerator.cpp`**
    - Main workflow (200 lines)
    - MQTT configuration
    - Bus type selection
    - Output handling
7. **`src/ConfigGenerator_Mappings.cpp`**
    - Mapping CRUD operations (220 lines)
    - Interactive mapping management
8. **`src/ConfigGenerator_Dbus.cpp`**
    - D-Bus field prompts with introspection (280 lines)
    - Service/path/interface/signal/method prompts
    - Bus type implications
9. **`src/ConfigGenerator_Utils.cpp`**
    - YAML generation (150 lines)
    - File I/O
    - Error fixing

Modified Files

10. **`include/CLI.h`** - Added `CLIMode::GENERATE_CONFIG`
11. **`src/CLI.cpp`** - Parse `--generate-config` flag
12. **`src/main.cpp`** - Handle config generation mode
13. **`CMakeLists.txt`** - Add ncurses and new source files
14. **`debian/control`** - Add libncurses-dev dependency

Dependencies Added

Build dependency:

- `libncurses-dev` - For interactive TUI

Runtime dependency:

- `libncurses6` - Ncurses library

Total code added: ~1,400 lines of C++

Usage Examples

Example 1: Fresh Config

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

=== D-Bus to MQTT Bridge - Configuration Generator ===

--- MQTT Configuration ---
Enter MQTT broker hostname or IP [localhost]: mqtt.example.com
Enter MQTT port [1883]:
Enable MQTT authentication? [Y/n]: n

--- D-Bus Configuration ---
[Interactive selection of system/session]

--- Mappings Configuration ---
[Add mappings with introspection support]

--- Validating Configuration ---
✓ Configuration is valid!

--- Save Configuration ---
Enter output path (or press Enter for stdout): config.yaml
✓ Configuration saved to: config.yaml
```

Example 2: Fix Broken Config

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

Loading existing configuration from broken.yaml...
Loaded existing configuration. Will prompt for missing/invalid fields.

[Only prompts for invalid fields, uses valid ones as defaults]

--- Validating Configuration ---
✓ Configuration is valid!

✓ Configuration saved to: fixed.yaml
```

Example 3: D-Bus Service Discovery

```
Enter D-Bus service name:
  Type 'list' to browse available services
Service: list

[Ncurses TUI with arrow key navigation]
Select D-Bus Service (↑/↓ to navigate, Enter to select):

=== SYSTEM BUS ===
> [SYS] org.freedesktop.NetworkManager
  [SYS] org.freedesktop.systemd1
  [SYS] org.freedesktop.login1
  ...

=== SESSION BUS ===
  [SES] org.freedesktop.Notifications
  [SES] org.gnome.Shell
  ...

[Press Enter on selected service]

⚠ Note: org.freedesktop.systemd1 is a SYSTEM bus service
         Your config will need:
         - bus_type: system
         - Config saved to: /etc/dbus-mqtt-bridge/config.yaml
         ...
```

Building

```bash
# Install ncurses development package
sudo apt-get install libncurses-dev

# Build
./fix-packaging-issues.sh
dpkg-buildpackage -us -uc -b

# Install
sudo dpkg -i ../dbus-mqtt-bridge_0.1.0-1_amd64.deb
```

Testing

Manual Test 1: Interactive Generation

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

Follow prompts, test:

- ✓ MQTT broker/port prompts
- ✓ Authentication optional
- ✓ Bus type selection
- ✓ Mapping management
- ✓ Output to file

Manual Test 2: D-Bus Introspection

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

When prompted for service:

- Type `list` → should show ncurses TUI
- Navigate with arrows → selection highlights
- Press Enter → should select service
- Type `introspect` for signals → should list available

Manual Test 3: Fix Broken Config

```bash
# Create broken config
cat > broken.yaml <<EOF
mqtt:
  broker: "invalid hostname!"
  port: 99999
bus_type: invalid
EOF

# Fix it
dbus-mqtt-bridge --generate-config --from broken.yaml -o fixed.yaml
```

Should:

- ✓ Load broken config
- ✓ Show validation errors
- ✓ Prompt to fix
- ✓ Re-prompt for invalid fields
- ✓ Save valid config

Manual Test 4: Partial Config

```bash
# Create partial config (only MQTT section)
cat > partial.yaml <<EOF
mqtt:
  broker: localhost
  port: 1883
EOF

# Complete it
dbus-mqtt-bridge --generate-config --from partial.yaml
```

Should:

- ✓ Use localhost/1883 as defaults
- ✓ Skip those prompts (already valid)
- ✓ Prompt for bus_type
- ✓ Prompt for mappings

Known Limitations

1. **Terminal requirements:** Needs terminal that supports ncurses (most modern terminals)
2. **Introspection errors:** If D-Bus service doesn't support introspection, fallback to manual entry
3. **Large lists:** Services with 100+ entries are navigable but may be slow on very old hardware
4. **No mouse support:** Keyboard only (by design - more universal)

Benefits

1. **User-friendly:** No need to manually write YAML
2. **Discoverable:** See what D-Bus services/signals exist
3. **Validating:** Catches errors before saving
4. **Fixable:** Can repair broken configs
5. **Scriptable:** Partial configs enable automation
6. **Secure:** Never scripts credentials

Next Steps

Phase 3.5 complete! Ready for:

- **Phase 4**: Man page documentation