User Guide & The Mycelium Directive Engine

User Guide & The Mycelium Directive Engine

Mycelium is more than just a software-defined radio; it is a **programmable RF automation platform**. This guide explains how to leverage its modular architecture and the "Directive Engine" to build complex, automated wireless missions.

The Mycelium Workflow

The power of Mycelium lies in its three-layer abstraction:

1. **Hardware (SDR)**: The physical or virtual interface to the radio spectrum.
2. **Tool (The Radio)**: A software-defined instance combining a specific **Protocol** (e.g., ACARS, IFF) and **Modulation** (e.g., AM, FM, MSK).
3. **Directives (The Brain)**: The logic that dictates how the Tool interacts with data, enabling fully autonomous operation.

Understanding the Directive Engine

The Directive Engine follows a **Condition Action AfterAction** logic flow. This allows you to create event-driven "if-this-then-that" behaviors for your radio tools.

1. Conditions: The "When"

Conditions monitor the tool's data buffer or internal state to determine when to act.

- **Null**: Always triggers (ideal for loops or continuous polling).
- **Any**: Triggers as soon as the buffer contains data.
- **Wait**: Triggers after a specified time delay (in milliseconds).
- **Value_Match**: Triggers when a specific variable or buffer segment matches a value.

2. Actions: The "What"

Actions perform the core tasks of the mission.

- **Receive_Audio / Receive_IQ**: Pull data from the SDR into the tool's buffer.
- **Transmit**: Push the tool's current buffer to the SDR for transmission.
- **File_Write / File_Read**: Log data or load signal samples from a file.
- **Extract_Variable**: Parse binary fields from a received packet into a named variable.
- **Insert_Variable**: Inject a variable (like a timestamp or custom ID) into a packet before transmission.
- **Print**: Output formatted strings (with variable expansion) to the console for monitoring.

3. AfterActions: The "Next"

AfterActions manage the lifecycle of the mission and allow for complex state transitions.

- **Stop_Tool**: Immediately halt the current tool.
- **Disable_Directive**: Turn off a specific logic branch once it has completed its task.
- **Jump**: Force the execution to jump to a different directive, enabling the creation of complex state machines.

The Power of Variable Expansion

Mycelium features a deep variable system that allows for dynamic behavior without the need to modify any code.

Built-in Variables

- `$timestamp`: The current system time.
- `$frequency`: The tool's current center frequency.
- `$data_hex`: The current buffer content in hexadecimal format.
- `$data_string`: The current buffer content as an ASCII string.

Custom & Extracted Variables

Extract specific bits from a protocol (like an aircraft's registration or tail number) and use them in subsequent actions:

```
# Extract the 7-character aircraft ID from offset 10 of an ACARS message
>> directive create -t acars_rx -n get_tail -c Any -a Extract_Variable -v TAIL_ID
 ↪  -o 10 -l 7

# Dynamically log the message to a file named after that specific aircraft
>> directive create -t acars_rx -n log_tail -c Any -a File_Write -f
 ↪  "logs/${TAIL_ID}.txt"
```

Practical Mission Example: Automated Responder

Build a system that listens for a specific "Ping" on 433 MHz and automatically replies with a "Pong" and a timestamp.

1. **Configure the Tool**:

   ```
   # Create a tool using the ISM_Generic protocol on 433.92 MHz
   tool create -p ISM_Generic -n responder tool set -n responder -r hackrf0 -f
    ↪  433920000 -m PWM
   ```

2. **Define the Detection Logic**:

   ```
   # Continuously poll the spectrum for 1024 samples
   >> directive create -t responder -n poll -c Null -a Receive_Audio -s 1024

   # Check if the received data matches our "PING" hex pattern
   >> directive create -t responder -n check_ping -c Value_Match -v "0xABCD" -a
    ↪  Print -F "Ping received!"
   ```

3. **Define the Response Logic**:

   ```
   # Prepare the response buffer with a dynamic timestamp
   >> directive create -t responder -n prep_pong -c Any -a Insert_Variable -v
    ↪  "PONG_$timestamp"

   # Transmit the response back into the spectrum
   >> directive create -t responder -n send_pong -c Any -a Transmit
   ```

Conclusion

By combining **Mission Packs** with the **Directive Engine**, Mycelium transforms from a simple SDR tool into a fully autonomous signal intelligence and response platform.
