Path Scripting

Contents

1 Using a Path Script

A **Path Script** allows OutpostX to connect to a BBS through an intermediate packet node or other system that requires one or more commands before the destination BBS is reached.

For example, a direct connection might simply connect your TNC to the BBS:

```
C K6FB
```

However, another station may require connecting first to an intermediate node and then issuing commands at that node to reach the BBS. A Path Script automates those additional steps.

2 Configuring a Path Script

Open **Setup > BBS**, select the desired BBS configuration, and select **Node** as the Path type.

Enter the required commands in the **Path Script** field. The script is executed from top to bottom during a Send/Receive session.

OutpostX currently recognizes four types of script lines:

```
# comment
TIMEOUT seconds
SEND "text"
WAITFOR OK "success" FAIL "failure"
```

Blank lines and lines beginning with # are ignored. The BBS Setup screen also provides a **Validate Script** function that checks the script format before it is used.

3 Path Script Commands

SEND - transmits the specified text to the connected system.

```
SEND "C ROCK"
```

The command is sent, but OutpostX does not automatically wait for a response. Use WAITFOR when the next step depends upon receiving a particular response.

WAITFOR - waits for either a successful response or a failure response.

```
WAITFOR OK "Help ?" FAIL "*** RET", "*** DISCONNECTED"
```

If an OK string is received, script execution continues with the next line. If one of the FAIL strings is received, the Path Script fails and the Send/Receive session is terminated.

More than one response may be specified:

```
WAITFOR OK "Help ?", "NODE>" FAIL "*** RET", "*** DISCONNECTED"
```

This is useful when different versions or configurations of a node may return slightly different responses. The current validator explicitly permits multiple quoted strings in both the OK and FAIL portions.

TIMEOUT - changes how long subsequent WAITFOR commands will wait for a response.

```
TIMEOUT 30
```

This can be useful for RF paths where establishing the next connection may take longer than usual.

Comments - begin with # and can be used to document the purpose of a script.

```
# Connect through the ROCK node to the BBS
```

4 Example, Single Hop

The following example connects to the ROCK node and then enters its BBS:

```
# Connect to the ROCK node
TIMEOUT 10
SEND "C ROCK"
WAITFOR OK "Help ?" FAIL "*** RET", "*** DISCONNECTED"

# Enter the BBS
SEND "BBS"
```

This is also essentially the example built into the current BBS Setup screen. The sequence is:

1. Set the response timeout to 10 seconds.
2. Send C ROCK.
3. Wait for Help ?, indicating that the connection succeeded.
4. Abort if *** RET or *** DISCONNECTED is received instead.
5. Send BBS to enter the destination BBS.

After the Path Script completes, OutpostX continues with its normal BBS connection processing.

5 Example, Multi-Hop Paths

A Path Script can contain multiple SEND and WAITFOR pairs, allowing OutpostX to traverse several nodes.

For example:

```
# Connect to first node
TIMEOUT 20
SEND "C NODE1"
WAITFOR OK "NODE1>" FAIL "*** RET", "*** DISCONNECTED"

# Connect from NODE1 to NODE2
SEND "C NODE2"
WAITFOR OK "NODE2>" FAIL "*** RET", "*** DISCONNECTED"

# Enter the BBS
SEND "BBS"
```

Each WAITFOR confirms that the previous step succeeded before the next command is sent. This is preferable to simply inserting delays because OutpostX advances only after receiving an expected response.

6 What Happens if the Script Fails?

A Path Script stops if a WAITFOR command times out or receives one of its FAIL responses. OutpostX reports the Path Script failure, aborts the upstream connection, and terminates the Send/Receive session rather than attempting normal BBS operations over an incomplete path. Operator cancellation is handled similarly.

The **Transcript** tab in the Send/Receive window is particularly useful when developing a Path Script. It shows the commands sent by OutpostX and the responses returned by the TNC, node, and BBS. Those responses can then be used to determine appropriate OK and FAIL strings for the script.