

## Using PDI Scripts - Part 1

Primitive data injection (PDI) injects primitive ordered sets or command data into the Fibre Channel data stream and monitors the results. PDI is implemented using PDI script language command words. PDI scripts can be developed to perform specific injection and monitoring. Scripts are developed using the PDI script editor and PDI script language. These scripts, which must conform to the requirements of the PDI script language, can be saved as files and compiled with the PDI script compiler. The binary file created from the PDI script compiler can be downloaded to the IFC-4 Fibre Channel Tester where it is then executed.

### Sample PDI Scripts

#### enter\_oldport.pdi

This script demonstrates the correct way to switch from arbitrated loop to a point-to-point topology when connected to a switch. The condition statements provide various exit strategies from the script. When a script is compiled in PDI, another text file is created and displayed in EAGLE™ software. This file is the hardware address equivalent of the script. When the compiled test is executed, the hardware addresses provide a map that points the user to the test's progress based on where it exited the script.

```
// LIP and then NOS for 1*AL_TIME
SEND LIP F7 F7 COUNT=3
SEND NOS TIME=15

// continue sending NOS until OLS or NOS back;
// up to 4 sec
WAIT UNTIL TIME=4000 OR NOS GOTO TAG 2
WAIT UNTIL TIME=1 OR OLS GOTO TAG 3
TAG 1
SEND IDLE TIME=2000
HALT // timeout

// got NOS back
TAG 2
SEND OLS TIME=1
WAIT UNTIL TIME=2000 GOTO TAG 1 OR LR
// 2-sec timeout
SEND LRR TIME=1
SEND IDLE TIME=2000 // get past a possible ELP
HALT

// got OLS back
TAG 3
SEND LR TIME=1
WAIT UNTIL TIME=2000 GOTO TAG 1 OR LRR
```

```
// 2-sec timeout
SEND IDLE TIME=2000 // get past a possible ELP
HALT
```

### myloopint.pdi

This script is an example of a Fibre Channel loop initialization process. The Fibre Channel frames were scripted in an external text editor and then compiled with the PDI.

```
// myloopinit.pdi -

SEND LIP COUNT=13
ECHO UNTIL LIP
SEND IDLE

// Send LISM frames 2.5 Msec apart with a high
//priority WWN
SEND IDLE TIMEMICRO=2500
TAG 1
SEND SOFI3
SEND 220000EF
SEND 000000EF
SEND 01380000
SEND 00000000
SEND FFFFFFFF
SEND 00000000
SEND 11010000
SEND 10000000
SEND 00000002
SEND EOFT
SEND IDLE

SEND IDLE UNTIL TIMEMICRO=2500 GOTO TAG 1 OR SOFI3

// have gotten sofi3, look for LISM with
// our forwarded WWN
WAIT UNTIL 00000002 POSITION=9 GOTO TAG 2
// must have goto

// not our WWN, back to tag 1
GOTO TAG 1

// got LISM with our WWN, send ARB F0 and wait
//for it back
TAG 2
SEND ARB F0 UNTIL BC94F0F0 FLAGS=K000

// start LIXA frame exchange as master
SEND SOFi3 //LIFA frame
SEND 220000EF
SEND 000000EF
```

```
SEND 01380000
SEND 00000000
SEND FFFFFFFF
SEND 00000000
SEND 11020000
SEND 00000000
SEND 00000000
SEND 00000000
SEND 00000000
SEND 00000000
SEND EOFt
//SEND IDLE COUNT=4000
```

```
SEND IDLE UNTIL SOFi3
SEND IDLE COUNT=4000
```

```
SEND SOFi3           //LIPA frame
SEND 220000EF
SEND 000000EF
SEND 01380000
SEND 00000000
SEND FFFFFFFF
SEND 00000000
SEND 11030000
SEND 20000000
SEND 00000000
SEND 00000000
SEND 00000000
SEND EOFt
//SEND IDLE COUNT=4000
```

```
SEND IDLE UNTIL SOFi3
SEND IDLE COUNT=4000
```

```
SEND SOFi3           //LIHA frame
SEND 220000EF
SEND 000000EF
SEND 01380000
SEND 00000000
SEND FFFFFFFF
SEND 00000000
SEND 11040000
SEND 20000000
SEND 00000000
SEND 00000000
SEND 00000001
SEND EOFt
//SEND IDLE COUNT=4000
```

```
SEND IDLE UNTIL SOFi3
SEND IDLE COUNT=4000
```

```
SEND SOFi3           //LISA frame
```





```
SEND FFFFFFFF
SEND FFFFFFFF
SEND EOFt
//SEND IDLE COUNT=4000
```

```
SEND IDLE UNTIL SOFi3
SEND IDLE COUNT=2000
SEND CLS
SEND IDLE UNTIL CLS
```

```
HALT
```