Since its debut in December 1999, Linux on S/390 and zSeries has evolved with great speed. Device support has been a major area of development. A major milestone for device support was reached with the addition of SCSI over fibre support.

This support was first demonstrated to the public with the almost bizarre exhibition at LinuxWorld 2002 in New York of a z900 burning CDs. Such an exhibition was good theatre as it generated great interest from the mainstream.

Software AG was approached to take part in an early field test of this support in July 2002. For this test, however, nothing as esoteric as CD burning would be done, just regular disk operations using our IBM Shark and EMC Symmetrix subsystems.

This report documents the experiences of installing, configuring and using this support. Before getting into the “guts” of the matter I’ll get ahead of myself and state that it is a testament to the underling SCSI code and that developed by IBM that we were up and running on the first day.
Thanks

- Software AG began participation in the ESP in July, 2002
- IBM have authorized Software AG to present this material
- Thanks for their assistance during the ESP goes to:
  - Ingolf Salm – IBM Germany
  - Pam Hares – IBM UK
  - Christoph Arenz – IBM Germany (& US)
Acknowledgements

• Uli Kuhna – for SAN and Shark configuration
• Wolfgang Buettner – for z/VM & IOCDS configuration
• Dr. Gerhard Banzhaf (IBM) – material from his presentation “FCP Channel for z800 and z900”
• Material adapted from “Device Drivers and Installation Commands”.
• Material sourced from “Getting Started with zSeries Fibre Channel Protocol”.
Agenda

- Background
- Environment
- Configuration
- 31-bit Experiences
- 64-bit Experiences
- Further work
**Background**

- Allows Linux for S/390 (31-bit mode) and Linux for zSeries (64 bit mode) to access:
  - Distributed storage devices
    - With FCP interfaces (via switches)
    - With parallel SCSI interfaces (via additional bridges)
  - With Linux for zSeries (S/390) running
    - In a partition
    - Under z/VM (requires z/VM 4.3 RSU001)
- Provides access to distributed (open) storage and SAN world
- Based on existing FICON and FICON Express hardware
Background

- New CHPID type: FCP
- Uses 2-port Fibre Channel cards FICON and FICON Express
  - Optical only
  - Short wave and long wave
  - 1 Gbit/s today: 2 Gbit/s has been announced
  - Currently 232KB buffer in card: 2MB proposed
- Different Firmware Load
  - Selected via definition of CHPID type in IOCP (HCD)
- QDIO protocol for communication between Processor/Memory and Channel
  - Based on scheme introduced with OSA Express
  - Continuously running channel programs
    - Reduces I/O path lengths
    - Reduces number of interrupts
Thezfcp driver is a low-level or host-bus adapter driver supplementing the Linux SCSI I/O subsystem (SCSI stack).

• zfcp driver is open source.

• Linux for zSeries and S/390 can make use of all SCSI device types currently supported by Linux on other platforms including
  – SCSI disks,
  – Tapes,
  – CD-ROMs, and,
  – DVDs.

Thezfcp driver is a low-level or host-bus adapter driver supplementing the Linux SCSI I/O subsystem (SCSI stack). Thus, Linux for zSeries and S/390 can make use of all SCSI device types currently supported by Linux on other platforms including SCSI disks, tapes, CD-ROMs, and DVDs. Both ESA (31 bit Linux) and ESAME (64 bit Linux) are supported.
Background

Linux SCSI Stack unchanged

Filesystem
- block devices: DASD, CDROM, DVD
- character devices: CD Writer, Scanner

Common SCSI Stack
- FCP: zSeries FCP adapter device driver

Replaces adapter specific device drivers
SAN Topologies

- **Point-to-point**
  
  Point-to-point: This is the simplest topology to configure. A point-to-point configuration is a direct connection between two endpoints. Typically, it consists of a host, a device (such as a disk controller), and a dedicated fibre link.

- **Arbitrated Loop**
  
  Arbitrated Loop: This is a ring topology that shares the fibre channel bandwidth among multiple endpoints. The loop is implemented within a hub that interconnects the endpoints.

  An arbitrated scheme is used to determine which endpoint gets control of the loop. The maximum number of ports is 127.
Switched Fabric

This topology provides the most flexibility and makes the best use of the aggregated bandwidth by the use of switched connections between endpoints. One or more switches are interconnected to create a fabric, to which the endpoints are connected.
## Environment

- **Hardware**

<table>
<thead>
<tr>
<th>Component</th>
<th>Model</th>
<th>Microcode Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM z900</td>
<td>2064-103</td>
<td>Level 8 (ML8)</td>
</tr>
<tr>
<td>McData SAN Switch</td>
<td>Connectrix ED-64M</td>
<td>01.04.002</td>
</tr>
<tr>
<td>Shark ESS</td>
<td>2105-F20</td>
<td>1.5.0.107</td>
</tr>
<tr>
<td>EMC</td>
<td>8730</td>
<td></td>
</tr>
<tr>
<td>IBM Magstar</td>
<td>3590-E</td>
<td></td>
</tr>
</tbody>
</table>
## Environment

- Software

<table>
<thead>
<tr>
<th>Component</th>
<th>Function</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/VM</td>
<td>Hypervisor</td>
<td>4.3 RSU 002</td>
</tr>
<tr>
<td>Linux for S/390</td>
<td>31-bit kernel</td>
<td>2.4.7+</td>
</tr>
<tr>
<td>Linux for zSeries</td>
<td>64-bit kernel</td>
<td>2.4.17+</td>
</tr>
<tr>
<td>ESM Manager</td>
<td>SAN Manager</td>
<td>2.0</td>
</tr>
</tbody>
</table>
Environment

- Linux details
  - CONFIG_MSDOS_PARTITION=y
  - CONFIG_ZFCP=m
  - New utils-linux – fdisk required
  - Will (eventually) need new devs.rpm but not mandatory
  - Manuals geared towards devfs but we were using SLES7
SAN Configuration

**CHPID E0**
Dev: E000
WWPN: 50050764012006f6

**CHPID E1**
Dev: E100
WWPN: 50050764012006f6

**DAEV:** 5005076400c67100

**FC-Director**
DAEDIR01

**ESS 2105-F20**
- Target 0: 100GB
- Target 1: 10GB

**IBM 3590E**

**EMC 8730**
- Target 0: 5.8GB

**Zoning**
WWPN: 50050763004029a0
**z/VM User Directory**

**PROFILE LNXGST**

- MACHINE ESA
- OPTION QUICKDSP
- SHARE REL 1000
- IPL CMS PARM AUTOCK
- CONSOLE 009 3215 T OPERATOR
- SPOOL 00C 2540 READER *
- SPOOL 00D 2540 PUNCH A
- SPOOL 00E 1403 A
- LINK MAINT 190 190 RR
- LINK MAINT 190 190 RR
- USER DAEV005 ******** 256M 512M G
- INCLUDE LNXGST
- ACCOUNT LINUX R.WAITE
- MDISK 0191 3390 2651 0025 VVSYS1 MR
- MDISK 0192 3390 0001 2838 VVTL99 MR
- MDISK 0193 3390 2839 0500 VVTL99 MR
- SPECIAL E00A CTCA TCPIP
- SPECIAL E00B CTCA TCPIP
- HyperSockets with MTU size of 8k
- DEDICATE D10 D10
- DEDICATE D11 D11
- DEDICATE D12 D12

**USER DAEV006 ******** 256M 512M G**

- INCLUDE LNXGST
- ACCOUNT LINUX R.WAITE
- MDISK 0191 3390 2676 0025 VVSYS1 MR
- MDISK 0192 3390 3839 2838 VVTL99 MR
- MDISK 0193 3390 3339 0500 VVTL99 MR
- SPECIAL E00C CTCA TCPIP
- SPECIAL E00D CTCA TCPIP
- HyperSockets with MTU size of 8k
- DEDICATE D04 D04
- DEDICATE D05 D05
- DEDICATE D06 D06

Note: The SCSI devices ‘E000/E100’ were ATTACHed to each user as required:

FCP E000 ATTACHED TO DAEV006 E000 CHPID E0
• IOCDS

CHPID PATH=(E0), SHARED,
   PARTITION=({DAEV, DAEX, DALI}, {DAEV, DAEX, DALI}), TYPE=FCP
CNTLUNIT CUNUMBR=E0FC, PATH=(E0), UNIT=FCP
IODEVICE ADDRESS=(E000, 016), CUNUMBR=(E0FC), UNIT=FCP

CHPID PATH=(E1), SHARED,
   PARTITION=({DAEV, DAEX, DALI}, {DAEV, DAEX, DALI}), TYPE=FCP
CNTLUNIT CUNUMBR=E1FC, PATH=(E1), UNIT=FCP
IODEVICE ADDRESS=(E100, 016), CUNUMBR=(E1FC), UNIT=FCP
Configuration

- CHPID/Device/SCSI Device
  - S390 device ≠ SCSI device
  - S390 device is conduit to SCSI
  - May be 1,…,n SCSI devices at end of conduit
  - Each SCSI device may be partitioned to produce multiple targets
A new addressing scheme was developed for Fibre Channel Protocol (FCP) usage, built around World Wide Names (WWN) that are eight bytes long.

Part of the name represents an address type, part is a number that identifies the manufacturer, and part is a unique number assigned (by the manufacturer) for each port or node.

A node is typically a box that contains information. Nodes have one or more ports (and only FC ports are relevant here). A given box may have several addresses; one for the node itself, and one for each FC port contained in the node. Abbreviations are:

- WWNN is a World Wide Node Name
- WWPN is a World Wide Port Name
- WNN is any World Wide Name (WWNN or WWNP)
• Linux provides extensive support for SCSI devices, based on the more advanced forms of traditional SCSI addressing.

• A SCSI device is expected to have a number (target address, typically a single digit) and this target may have multiple LUNs.

• The full SCSI addressing scheme can be used:
  – Device number (from your IOCDS) (coded as a “host number”)
  – A bus number on this adapter (always zero in current zSeries implementations)
  – A SCSI target number on this bus
  – A LUN number on this SCSI target
  – A partition or device within this LUN
Configuration

- FCP Mapping

1. 0xe000
2. 0x01:0xpppppppppppppppp
3. 0xe000
4. 0x00:0xnnnnnnnnnnnnnnnn

LUN name within node
LUN number used by Linux
WWPN of SAN node
SCSI target number
IOCDS device number
1. Device number

- This device number must be defined in the IOCDS and be assigned to the FCP channel attached to the FCP switch.
  - You can use the same device number for all your FCP connections (by a given Linux image), or,
  - You can elect to use multiple device numbers (all assigned to the FCP channel, of course).
2. Target number

- The standard Linux SCSI support understands traditional SCSI addressing, with adapter numbers, bus numbers, target (“SCSI address”) numbers, and LUNs.
- You assign this number.
- Usable target addresses range from 1 to any positive 31-bit number.
- Traditional SCSI target addresses ranged from 0-7 or 0-15. More recent SCSI architecture allows a much larger number.
3. WWPN
   – The World Wide Port Name of the device containing the LUN.
   – The WWPN *as seen by the FC switches*. WWPNs reported by devices may be slightly different.
   – You do not assign this number; it is built into the FCP devices and you must determine the proper number by querying your FCP elements.
4. Linux LUN
   - The number *to be used by* Linux.
   - You assign this number.
   - Normal usage is to start with 0 and to increment by one for each LUN.

5. SAN Device LUN
   - The number used by the remote device.
   - You do not assign this number.
   - You must determine the number assigned by the node controller.
Configuration

- There are different parameters that can or must be supplied by the user to allow for proper zfcp operation:
  - Address mappings between Linux SCSI and FCP schemes (optional for each SCSI target)
  - Logging level to determine the verbosity of the zfcp device driver (optional, default value is used if not supplied)
Configuration

- The zfcp driver provides different means of configuration:
  - Kernel parameters
  - Module parameters (such as for use in modules.conf)
  - Various proc file system entries in /proc/scsi/zfcp

There are different parameters that can or must be supplied by the user to allow for proper zfcp operation:
- Address mappings between Linux SCSI and FCP schemes (optional for each SCSI target)
- Logging level to determine the verbosity of the zfcp device driver (optional, default value is used if not supplied)

For these purposes, the zfcp driver provides different means of configuration:
- Kernel parameters
- Module parameters (such as for use in modules.conf)
- Various proc file system entries in /proc/scsi/zfcp
### Configuration

- Module Parameters and proc File System entries

<table>
<thead>
<tr>
<th>Function</th>
<th>Module Parameter</th>
<th>Kernel Parameter</th>
<th>proc-fs entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set logging level</td>
<td>loglevel zfcp_</td>
<td>loglevel</td>
<td>/proc/scsi/zfcp/mod_parm</td>
</tr>
<tr>
<td>Get logging level (and other global module information)</td>
<td>N/A</td>
<td>N/A</td>
<td>/proc/scsi/zfcp/mod_parm</td>
</tr>
<tr>
<td>Add address mapping(s)</td>
<td>map</td>
<td>zfcp_map</td>
<td>/proc/scsi/zfcp/add_map</td>
</tr>
<tr>
<td>Get all existing address mappings</td>
<td>N/A</td>
<td>N/A</td>
<td>/proc/scsi/zfcp/map</td>
</tr>
</tbody>
</table>
The SLES7 Linux system we were using did not use the devfs file system nor did it have definitions in the /dev directory. To rectify this we used the commands found above. SLES8 has the nodes defined and does not require these commands to be issued.
# Configuration

- **ESS Devices**

<table>
<thead>
<tr>
<th>Target</th>
<th>Logical Unit</th>
<th>Size</th>
<th>Device Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>25GB</td>
<td>/dev/sda1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>50GB</td>
<td>/dev/sda2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>25GB</td>
<td>/dev/sda3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>25GB</td>
<td>/dev/sda4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>5GB</td>
<td>/dev/sdb1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5GB</td>
<td>/dev/sdb2</td>
</tr>
</tbody>
</table>
Configuration

- Driver load script
  
  ```bash
  #!/bin/sh
  rmmod zfcp
  modprobe qdio
  modprobe scsi_mod
  insmod zfcp loglevel=0 map="
  0xe000 0x01:0x5005076300c38c6d 0x00:0x5200000000000000;
  0xe000 0x01:0x5005076300c38c6d 0x01:0x5201000000000000"
  modprobe sd_mod
  modprobe st
  ```

Based on the example provided in the Redpaper “Getting Started with zSeries Fibre Channel Protocol” we created the above script to load the appropriate device drivers and provide the correct parameters for the ESS devices.
During scsi_load:

```
sfcp: FSF: sfcp_fsf_send_fcp_command_task_handler: status for SCSI Command:
00000000 00000000 00000202 00000000 00000000 00000000 00000000 00000000 00000000
zfcp: FSF: zfcp_fsf_send_fcp_command_task_handler: SCSI status code 0x2
00000000 00000000 00000000 00000202 00000000 00000000 00000000 00000000 00000000
zfcp: FSF: zfcp_fsf_send_fcp_command_task_handler: Attached scsi disk sda at scsi0, channel 0, id 1, lun 0
zfcp: FSF: zfcp_fsf_send_fcp_command_task_handler: status for SCSI Command:
```

SCSI device sda: 195312512 512-byte hdwr sectors (100000 MB)

```
sfcp: FSF: sfcp_fsf_send_fcp_command_task_handler: status for SCSI Command:
00000000 00000000 00000202 00000000 00000000 00000000 00000000 00000000 00000000
zfcp: FSF: zfcp_fsf_send_fcp_command_task_handler: Attached scsi disk sda at scsi0, channel 0, id 1, lun 0
zfcp: FSF: zfcp_fsf_send_fcp_command_task_handler: status for SCSI Command:
```

SCSI device sda: 195312512 512-byte hdwr sectors (100000 MB)
Configuration

- Following scsi_load:

<table>
<thead>
<tr>
<th>Module</th>
<th>Size</th>
<th>Used by</th>
</tr>
</thead>
<tbody>
<tr>
<td>st</td>
<td>27696</td>
<td>0 (unused)</td>
</tr>
<tr>
<td>sd_mod</td>
<td>12048</td>
<td>0 (unused)</td>
</tr>
<tr>
<td>zfcp</td>
<td>345312</td>
<td>0</td>
</tr>
<tr>
<td>scsi_mod</td>
<td>61344</td>
<td>3 [st sd_mod zfcp]</td>
</tr>
<tr>
<td>nfsd</td>
<td>69648</td>
<td>4 (autoclean)</td>
</tr>
<tr>
<td>qeth</td>
<td>153072</td>
<td>1 (autoclean)</td>
</tr>
<tr>
<td>qdio</td>
<td>33968</td>
<td>2 (autoclean) [zfcp qeth]</td>
</tr>
<tr>
<td>ipv6</td>
<td>247472</td>
<td>-1 (autoclean) [qeth]</td>
</tr>
<tr>
<td>8021q</td>
<td>12928</td>
<td>0 (autoclean) [qeth]</td>
</tr>
<tr>
<td>ctc</td>
<td>49840</td>
<td>1 (autoclean)</td>
</tr>
<tr>
<td>fsm</td>
<td>1920</td>
<td>0 (autoclean) [ctc]</td>
</tr>
</tbody>
</table>
• Partitioning using `fdisk`:

```
Disk /dev/sda: 255 heads, 63 sectors, 12157 cylinders
Units = cylinders of 16065 * 512 bytes

Device  Boot  Start  End   Blocks  Id  System
/dev/sda1  *   1   3188  25607578+   83  Linux
/dev/sda2   3189  6376  25607610   83  Linux
/dev/sda3   6377 12157  46435882+   83  Linux

Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

Our SLES7 system did not have the `fdisk` utility. However, thanks to our excellent working relationship with SuSE (and Bernd Kaindl in particular), we were able to acquire a newly built `utils-linux` package containing the necessary command. We used this command to divide the area of the SCSI device into several partitions as shown above.
• Mounting devices
  - ln -s /etc/init.d/scsi_mount /etc/init.d/rc3.d/S12scsimn

```bash
#!/bin/sh
mount /dev/sda1 /FS/scsi01
mount /dev/sda2 /FS/scsi02
mount /dev/sda3 /FS/scsi03
mount /dev/sdb1 /FS/scsi04
mount /dev/sdb2 /FS/scsi05
```
Configuration

- Unmounting devices
  - `ln -s scsi_unmount rc3.d/K13scsiun`

```bash
#!/bin/sh
umount /FS/scsi01
umount /FS/scsi02
umount /FS/scsi03
umount /FS/scsi04
umount /FS/scsi05
```
• Contents of `/proc/partitions`:

<table>
<thead>
<tr>
<th>major</th>
<th>minor</th>
<th>#blocks</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
<td>97656256</td>
<td>dasdbm</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>25607578</td>
<td>dasdbm1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>25607610</td>
<td>dasdbm2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>46435882</td>
<td>dasdbm3</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>9765632</td>
<td>dasdbn</td>
</tr>
<tr>
<td>8</td>
<td>17</td>
<td>5121008</td>
<td>dasdbn1</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>4643840</td>
<td>dasdbn2</td>
</tr>
<tr>
<td>94</td>
<td>0</td>
<td>360000</td>
<td>dasda</td>
</tr>
<tr>
<td>94</td>
<td>1</td>
<td>359988</td>
<td>dasdal</td>
</tr>
<tr>
<td>94</td>
<td>4</td>
<td>2043360</td>
<td>dasdb</td>
</tr>
<tr>
<td>94</td>
<td>5</td>
<td>2043348</td>
<td>dasdbl</td>
</tr>
</tbody>
</table>

➤ Suspect device naming due to IBM Partitioning being enabled
Linux provides a lot of data within the proc file system that allows you to inspect, monitor, and change the configuration and operation of the SCSI subsystem. Graphically, the sub-tree available looks like the above.
• Contents of `/proc/scsi/scsi`:

```
<table>
<thead>
<tr>
<th>Attached devices:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host: scsi0 Channel: 00 Id: 01 Lun: 00</td>
</tr>
<tr>
<td>Vendor: IBM Model: 2105F20 Rev: .107</td>
</tr>
<tr>
<td>Type: Direct-Access ANSI SCSI revision: 03</td>
</tr>
<tr>
<td>Host: scsi0 Channel: 00 Id: 01 Lun: 01</td>
</tr>
<tr>
<td>Vendor: IBM Model: 2105F20 Rev: .107</td>
</tr>
<tr>
<td>Type: Direct-Access ANSI SCSI revision: 03</td>
</tr>
<tr>
<td>Host: scsi0 Channel: 00 Id: 01 Lun: 00</td>
</tr>
<tr>
<td>Vendor: EMC Model: SYMMETRIX Rev: 5567</td>
</tr>
<tr>
<td>Type: Direct-Access ANSI SCSI revision: 02</td>
</tr>
</tbody>
</table>
```
**Configuration**

- **Contents of /proc/scsi/zfcp/dev0xe000/status:**

```plaintext
FCP adapter

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>device number</td>
<td>0xe000</td>
</tr>
<tr>
<td>registered on irq</td>
<td>0x0013</td>
</tr>
<tr>
<td>WWNN:</td>
<td>0x5005076400c67100</td>
</tr>
<tr>
<td>WWPN:</td>
<td>0x50050764012006f6</td>
</tr>
<tr>
<td>SW version:</td>
<td>0xe000</td>
</tr>
<tr>
<td>LIC version:</td>
<td>0xe0000001a</td>
</tr>
<tr>
<td>FC link speed</td>
<td>1 Gb/s</td>
</tr>
<tr>
<td>FC service class</td>
<td>3</td>
</tr>
<tr>
<td>SCSI host number</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Attached ports</td>
<td>2</td>
</tr>
<tr>
<td>Max SCSI ID of ports</td>
<td>0x00000001</td>
</tr>
<tr>
<td>Max SCSI LUN of ports</td>
<td>0x00000001</td>
</tr>
<tr>
<td>Scatter-gather table-size</td>
<td>57</td>
</tr>
<tr>
<td>Max no of queued commands</td>
<td>4096</td>
</tr>
<tr>
<td>Uses clustering</td>
<td>1</td>
</tr>
<tr>
<td>Uses New Error-Handling Code</td>
<td>1</td>
</tr>
<tr>
<td>ERP counter</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Adapter Status</td>
<td>0x54000000bf</td>
</tr>
</tbody>
</table>

**Adapter Structure information:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Magic</td>
<td>0xc6fcfcfc</td>
</tr>
<tr>
<td>Specific Magic</td>
<td>0xaaaaaaaaaa</td>
</tr>
<tr>
<td>Adapter struct at</td>
<td>0x109d82b0</td>
</tr>
<tr>
<td>List head</td>
<td>0x109d82b0</td>
</tr>
<tr>
<td>Previous list head</td>
<td>0x109d82b0</td>
</tr>
<tr>
<td>Port list head at</td>
<td>0x109d8840</td>
</tr>
<tr>
<td>Previous list head</td>
<td>0x109d8840</td>
</tr>
<tr>
<td>List lock</td>
<td>0x00000000</td>
</tr>
<tr>
<td>List lock owner PC</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>
```

---

**FCP driver $Revision: 3.60.4.1 $ (or for cryptography's sake 0x0003003c)**

- **device number:** 0xe000
- **registered on irq:** 0x0013
- **WWNN:** 0x5005076400c67100
- **WWPN:** 0x50050764012006f6
- **SW version:** 0xe000
- **LIC version:** 0xe0000001a
- **FC link speed:** 1 Gb/s
- **FC service class:** 3
- **SCSI host number:** 0x00000000

**Attached ports:** 2

**Max SCSI ID of ports:** 0x00000001
**Max SCSI LUN of ports:** 0x00000001
**Scatter-gather table-size:** 57
**Max no of queued commands:** 4096

**Uses clustering:** 1
**Uses New Error-Handling Code:** 1
**ERP counter:** 0x00000000
**Adapter Status:** 0x54000000bf
Configuration

• /proc/scsi/zfcp/dev0xe000/status (cont.):

O-FCP req list head: 0x0d0e250d0
Next list head: 0x0d0e250e08
List lock: 0x00000000 List lock owner PC: 0x00000000
Request queue at: 0x0d0e250f8
Free index: 002 Free count: 128
List lock: 0x00000000 List lock owner PC: 0x00000000
current TOD: 13263832012677040160
time lock held: 3507809781
Response queue at: 0x0d0e25550
Free index: 073 Free count: 000
List lock: 0x00000000 List lock owner PC: 0x00000000

DEVICE INFORMATION (devinfo):
Status: "OK"
Control Unit Type: 0x1731 Control Unit Model: 0x03
Device Type: 0x1732 Device Model: 0x03
CIWs: 0x40720080 0x41830004 0x42820040 0x431b1000
  0x441f0000 0x00000000 0x00000000 0x00000000

DEVICE INFORMATION (devstat):
Interrupt Parameter: 0x00000002 Last path used mask: 0x00
Channel Status: 0x00 Device Status: 0x00
Flag: 0x00000204 CCW address (from irb): 0x0d863a40
Response count: 0x00000000 Sense Count: 0x00000000
IRB: 0x00c0c0c9 0x0d863a40 0x00800000 0x00000000
Sense Data: 0x00000000 0x00000000 0x00000000 0x00000000
**Configuration**

- Contents of .../dev0xe000/id0x01/status:

<table>
<thead>
<tr>
<th>Port Information:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WWNN:</td>
<td>0x0000000000000000</td>
</tr>
<tr>
<td>WWPN:</td>
<td>0x5005076300c38c6d</td>
</tr>
<tr>
<td>SCSI-ID:</td>
<td>0xe00000001</td>
</tr>
<tr>
<td>Max SCSI lun:</td>
<td>0xe00000001</td>
</tr>
<tr>
<td>D-ID:</td>
<td>0x00614013</td>
</tr>
<tr>
<td>Handle:</td>
<td>0x00000025</td>
</tr>
<tr>
<td>Attached units:</td>
<td>2</td>
</tr>
<tr>
<td>ERP counter:</td>
<td>0xe0000000</td>
</tr>
<tr>
<td>Port Status:</td>
<td>0x54000003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port Structure information:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Magic:</td>
<td>0xfcfcfcfc</td>
</tr>
<tr>
<td>Specific Magic:</td>
<td>0xbbbbbbbb</td>
</tr>
<tr>
<td>Port struct at:</td>
<td>0xe0d869800</td>
</tr>
<tr>
<td>List head at:</td>
<td>0xe0d869808</td>
</tr>
<tr>
<td>Next list head:</td>
<td>0xe0d869808</td>
</tr>
<tr>
<td>Previous list head:</td>
<td>0xe0da250a0</td>
</tr>
<tr>
<td>Unit list head at:</td>
<td>0xe0d869820</td>
</tr>
<tr>
<td>Next list head:</td>
<td>0xe0d869508</td>
</tr>
<tr>
<td>Previous list head:</td>
<td>0xe0d869208</td>
</tr>
<tr>
<td>List lock:</td>
<td>0xe0000000</td>
</tr>
<tr>
<td>List lock owner PC:</td>
<td>0xe0000000</td>
</tr>
<tr>
<td>Parent adapter at:</td>
<td>0xe0da25000</td>
</tr>
</tbody>
</table>
**Configuration**

- Contents of `../id0x01/lun0x0/status`:

<table>
<thead>
<tr>
<th>Unit Information:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI lun:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>FCP lun:</td>
<td>0x5200000000000000</td>
</tr>
<tr>
<td>Handle:</td>
<td>0x00000001d</td>
</tr>
<tr>
<td>ERP counter:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Unit Status:</td>
<td>0x54000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Unit Structure information:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Magic:</td>
<td>0xfcfcfcfcfc</td>
</tr>
<tr>
<td>Specific Magic:</td>
<td>0xcccccccc</td>
</tr>
<tr>
<td>Unit struct at:</td>
<td>0x0d869500</td>
</tr>
<tr>
<td>List head at:</td>
<td>0x0d869508</td>
</tr>
<tr>
<td>Next list head:</td>
<td>0x0d869208</td>
</tr>
<tr>
<td>Previous list head:</td>
<td>0x0d869820</td>
</tr>
<tr>
<td>Parent port at:</td>
<td>0x0d869800</td>
</tr>
<tr>
<td>SCSI dev struct at:</td>
<td>0xe9f9e00</td>
</tr>
</tbody>
</table>
31-bit Experiences

- Vanilla kernel not sufficient:
  - fdisk partitioning “forgotten”
  - Problems with mke2fs:

```
# daev005:/usr/src/linux/drivers/s390/scsi
mke2fs /dev/sda2
mke2fs 1.19, 13-Jul-2000 for EXT2 FS 0.5b, 95/08/09
mke2fs: No such device or address while trying to
determine filesystem size
```

- Required reconfiguration:
  - File Systems->Partition Types->MSDOS

With the apparently successful partitioning of the device we attempted to use the mke2fs command to create an ext2fs on the various partitions. However, it was at this point we began running into trouble. The first partition appeared to work but the file system appeared to occupy the entire device not just the first partition (24414063 is the device size not the partition size).

An attempt to make a file system on the second partition did not work at all:
```
# daev005:/usr/src/linux/drivers/s390/scsi
mke2fs /dev/sda2
mke2fs 1.19, 13-Jul-2000 for EXT2 FS 0.5b, 95/08/09
mke2fs: No such device or address while trying to
determine filesystem size
```

We then specified loglevel=3 on insmod zfcp command to log debugging information. The log shows no activity when references to sda2/sda3 are made but lots of activity for sda1.

In order to determine if the error was peculiar to the 100GB device, we arranged with Software AG IT services to define another small SCSI device. We again used fdisk to split this 10GB device into two 5GB partitions. However, the same errors were encountered. Indeed, we experienced errors during load of the device drivers.

Following advice from Stefan Bader of IBM, we updated the Linux kernel configuration to enable support for: File Systems->Partition Types->MSDOS. After rebooting to pick up this new kernel, all our partitions became accessible.
Initially my insmod parameter was as follows:

```
0xe100 0x01:0x500604843e0c23eb
0x00:0x0000000000000001
```

After (apparently) successfully accessing the disk I attempted to partition it using `fdisk`. The utility reported success, but `syslog` messages were generated that indicated this was not the case.
I created a development environment for our Adabas and Adabas SQL Access (AQA) products, and proceeded to build them. The build involved significant amounts of compilation and linking that would allow me to exercise the disks. No problems were encountered during the build processes.

I created 2 Adabas databases on the Linux guest DAEV005 using the SCSI disks. One was for the normal installation verification databases and one for use by AQA and has sizes of:

- ASSO - 100MB
- DATA - 100MB
- WORK - 60MB

I installed AQA on DAEV005, built our internal test suite, and ran the various test scripts. In addition I rebuilt the AQA package using the SCSI area as the build area. No problems experienced.
31-bit Experiences

- Simplistic benchmarking
  - Used `dd` to exercise disk writes
  - Changed the underlying file system
    - ext2
    - ext3
    - reiser
- Uncontrolled environment & unscientific methodology
  - CPUs shared with other LPARs
  - LPAR has low weighting
- Results are indicative only

I performed simple experiments on the disks using the script found below to create a file half the size of the disk area. On DAEV005 I used the EMC devices, on DAEV006 I used the ESS devices. The testing was done in an uncontrolled environment: running on a lowly weighted LPAR sharing 1 of 3 processors with several other LPAR-based z/OS and VSE systems.

```rexx
#!/usr/bin/rexx
/* */
trace o
fs.1 = 'mke2fs'
fs.2 = 'mkreiserfs'
fs.3 = 'mke2fs -j'
fs.0 = 3
id.1 = 'ext2fs'
id.2 = 'reiserfs'
id.3 = 'ext3fs'
op.1 = ''
op.2 = '<tstfs.in'
op.3 = ''
bs.1 = 512; bs.2 = 1024; bs.3 = 4096; bs.4 = 8192; bs.5 = 16384; bs.0 = 5
Target = '/FS/scsi04' /* '/FS/tempfs' on DAEV005 */
FCPdev = '/dev/sdb1' /* '/dev/sda2' on DAEV005 */
'echo y >tstfs.in'
Capacity = 'df'(Target)
parse var Capacity 'Mounted on' . Capacity .
Capacity = Capacity % 2
do I_fs = 1 to fs.0
  'umount' Target
  fs.I_fs FCPdev '>/dev/null 2>&1' op.I_fs
  'mount' FCPdev Target
  say COPIES('-',60)
  do I_bs = 1 to bs.0
    say '=>' bs.I_bs
    do I_Task = 1 to 4
      count = (Capacity * (1024 / bs.I_bs)) % 1
      say id.I_fs '('count')'
      'time dd if=/dev/zero of=Target/dummy.file',
      'bs=bs.I_bs 'count=COUNT
      'sleep 2s'
      'time rm' Target'/dummy.file'
      'sleep 2s'
    end
  end
end
exit
```
31-bit Experiences

dd if=/dev/zero of=/FS/tempfs/dummy.file bs=n count=nnnnnnn
31-bit Experiences

EMC - File System Write Performance

dd if=/dev/zero of=/FS/tempfs/dummy.file bs=n count=n
31-bit Experiences

EMC-File System Erase Performance

Time (s)

File System

ext2fs
reiserfs
ext3fs

Block Size

16kB
8kB
4kB
1kB
512B

rm /FS/tempfs/dummy.file
64-bit Experiences

- Product Testing
  - Adabas & AQA as per 31-bit
  - Included Natural
  - All worked as expected
- Similar transfer rate found when doing simplistic “dd” test

The same process was carried out for our 64-bit Linux system. The kernel was rebuilt, device nodes created, and device driver loaded. The startup and shutdown scripts created for the 31-bit system were reused in the 64-bit system. The devices were successfully sensed, drivers loaded and file systems mounted.

I created 2 Adabas databases on the Linux guest DAEV006 using the SCSI disks. One was for the normal vehicles etc. db and one for use by AQA and has sizes of:

- ASSO - 100MB
- DATA - 100MB
- WORK - 60MB

I installed AQA on DAEV006, built the internal test suite, and ran the test scripts. In addition I rebuilt the AQA package using the SCSI area as the build area. No problems experienced.

Activities included:

§ Generation of large databases;
§ Large tables and indexes were created, filled, interrogated and dropped;
§ Full logging was enabled to cause the production of large trace files (~2GB);
§ Natural was used to extract data from the Adabas database.

No problems were encountered.
dd if=/dev/zero of=/FS/scsi03/dummy.file bs=n count=nnnnnnn
64-bit Experiences

dd if=/dev/zero of=/FS/scsi03/dummy.file bs=n count=nnnnnnn

ESS - File System Write Performance

% Kernel Land

Block Size: 512B, 1kB, 4kB, 8kB, 16kB

File System: ext2fs, ext3fs, reiserfs, ext3fs

Kernel Land (%) for different block sizes and file systems.
64-bit Experiences

ESS - File System Erase Performance

File System

Time (s)

- ext2fs
- reiserfs
- ext3fs
- 512B

Block Size

- 16kB
- 8kB
- 4kB
- 1kB

rm /FS/scsi03/dummy.file
Microcode Upgrade

- Applied September 2002
- Increases buffer from 234K to 2MB
- Necessary for streaming tape
- Should improve big transfers in general

The previous tests were repeated after a microcode fix was applied to the 2064 to support 2MB buffers within the FCP card. Note the greatly improved data rate for Ext2 and Ext3.
ESS – Write Performance 2MB Buffer

ESS Write Performance - 2MB Buffer

File System

Block Size

512B

1K

4K

8K

16K

MB/s

0.00

5.00

10.00

15.00

20.00

25.00

30.00

35.00

40.00

45.00

ext2fs

reiserfs

ext3fs
ESS - Erase Performance 2MB Buffer

ESS Erase Performance - 2MB Buffer

File System
ext2fs
reiserfs
ext3fs

Block Size
512B
1K
4K
8K
16K

Time (s)
0
2
4
6
8
10
12
14
16
Tape – 3590E

- Loaned device
- Two new RPMs:
  - IBMTapeutil
    A tape utility program that exercises or tests the functions of the Linux device driver, IBMTape. It performs tape and medium changer operations.
  - IBMTapeconfig
    A script which creates and destroys IBMTape device special files according to the information logged in /proc/scsi/IBMTape and /proc/scsi/IBMchanger files.
**IBM Tape Utility**

---

### General Commands:
1. Open a Device
2. Close a Device
3. Inquiry
4. Test Unit Ready
5. Reserve Device
6. Release Device
7. Request Sense
8. Log Sense Page
9. Mode Sense Page
10. Switch Tape/Changer Device
11. Create Special Files
12. Query Driver Version
13. Quit IBMtapeutil

### Tape Commands:
20. Rewind
21. Forward Space Filemarks
22. Backward Space Filemarks
23. Forward Space Records
24. Backward Space Records
25. PSPM
26. BSFM
27. Space to End of Data
28. Read and Write Tests
29. Write Filemarks
30. Read or Write Files
31. Erase
32. Reset Drive
33. Set Block Size
34. Retention Tape
35. Query/Set Tape Position
36. Query Tape Status
37. Load Tape
38. Unload Tape
39. Lock Tape Drive Door
40. Unlock Tape Drive Door
41. Take Tape Offline
42. Enable/Disable Compression
43. Flush Driver's Buffer
44. Self Test
45. Display Message
46. Query Sense
47. Query Inquiry
48. Query/Set Tape Parameters
49. Query/Set Tape Position
50. Query/Set MT/ST Mode
51. Report Density Support
52. Locate Tape Position
53. Read Tape Position
54. Query Mcdevice Number
55. Synchronize Buffers
56. List Tape Filemarks
57. Dump Device
58. Load Ucode
59. Force Dump
60. Reset Drive
61. Take Tape Offline
62. Enable/Disable Compression
63. Flush Driver's Buffer
64. Self Test
65. Display Message
66. Query Sense
67. Query Inquiry
68. Query/Set Tape Parameters
69. Query/Set Tape Position
70. Locate Tape Position
71. Read Tape Position
72. Query Mcdevice Number
73. Synchronize Buffers
74. List Tape Filemarks
75. Dump Device
76. Load Ucode
77. Force Dump
78. Reset Drive
79. Take Tape Offline
80. Enable/Disable Compression
81. Flush Driver's Buffer
82. Self Test
83. Display Message
84. Query Sense
85. Query Inquiry
86. Query/Set Tape Parameters
87. Query/Set Tape Position
88. Locate Tape Position
89. Read Tape Position
90. Query Mcdevice Number
91. Synchronize Buffers
92. List Tape Filemarks
93. Dump Device
94. Load Ucode
95. Force Dump
96. Reset Drive
97. Take Tape Offline
98. Enable/Disable Compression
99. Flush Driver's Buffer
100. Self Test
101. Display Message

---

99. Back To Main Menu
IBM Tape Utility

Inquiry Data:
- Peripheral Qualifier: 0x00
- Peripheral Device Type: 0x01
- Removal Medium Bit: 1
- Device Type Modifier: 0x00
- ISO version: 0x00
- ECMA version: 0x00
- ANSI version: 0x03
- Asynchronous Event Notification Bit: 0
- Terminate I/O Process Message Bit: 0
- Response Data Format: 0x02
- Additional Length: 0x33
- Medium Changer Mode: 0x00
- Relative Addressing Bit: 0
- 32 Bit Wide Data Transfers Bit: 0
- 16 Bit Wide Data Transfers Bit: 0
- Synchronous Data Transfers Bit: 0
- Linked Commands Bit: 0
- Command Queuing Bit: 0
- Soft Reset Bit: 0
- Vendor ID: IBM
- Product ID: 03590E11
- Product Revision Level: E350
FCP Attached SCSI Tape

Block Size

MB/s

FCP attached SCSI Tape

Block Size

4096 8192 16384 32768 65536 131072 262144

0.00 5.00 10.00 15.00 20.00 25.00 30.00 35.00 40.00

SHARE