Problem Statement: SCSI Device Naming

rev. 0.99 13-Jun-1995

1 What problem are we trying to solve? Is it the right problem?

This Problem Statement looks at ways we might solve the problem of naming SCSI devices in a cluster. It defines the problem and lays out some broad guidelines for proposed solutions.

THE PROBLEM: How do we name devices so that each device in a cluster has a unique name?

Naming is important because guaranteed unique device names are key to sharing. "Sharing" is used in the multi-initiator sense -- independent access to a device from multiple nodes.

First it is fair to ask if this is the right Problem; if maybe there isn't some larger or smaller problem we should be solving instead. Perhaps we're working at the wrong level. In fact, there IS a "wider scope" problem that this statement does not address. Namely, that everything should work automatically. If we can't deliver that then the next-best solution is one that comes close. Whatever we produce ought to make customers think of Digital as problem-solvers.

This is made all the more important by the nature of our SCSI disks, which look like so many little green boxes innately indistinguishable from one another except perhaps for magic marker scribbles, affixed gummed labels or the like. Nevertheless a customer will want to "plug-and-play" disk drives. For example, VMS should somehow recognize when media are inserted onto -- or removed from -- a SCSI bus ("hot-swapping") and take the appropriate correct action. This might involve, among other things, mounting a disk clusterwide, adding a new member to a host-based shadowset or restricting a "secured" disk's access to a single process on a single node. It is important that the right thing get done and get done automatically, more so with SCSI than prior technologies precisely because SCSI disks are so primitive that human error is likely to be the rule rather than the exception. We need to eliminate human error, or at least minimize its likelihood and/or impact.

Regrettably, solving a problem of that magnitude would take huge amounts of time and resources. But one thing is clear: in order to even *start to work* on the above problem we will need to solve the Device Naming Problem. Without a solution to the device naming problem we can't possibly cause anything useful to happen automatically. Device naming is a fundamental building block; an "enabling technology" for the higher level capabilities described above.

Of special note, "hot-swapping" is not directly addressed in this Problem Statement. This Problem Statement proposes to name the slot rather than the brick. Thus one should be able to do hot-swapping by dismounting one disk and mounting a different disk, much the same as we would swap RA60s or other removable packs.

1.1 Device naming vs. data naming

It is fair to ask if "device naming" is the correct base technology, as opposed to "data naming". Customers don't care about "devices", they care about the <u>data</u> on those devices. Therefore, shouldn't the right approach involve the data rather than the device?

The answer is no. In fact this is just a simplified version of the previous argument. While the data *is* more important than the device, there is no way you can get along without a device name at the base of the technology pyramid. Obvious problem situations arise when a customer wants to use an old disk that has stale data on it, or INIT a new disk with nonstandard parameters. In VMS terms, there has to be a way to refer to a *unit* rather than a *volume*, because certain fundamental operations are necessarily done on a unit basis. This is exactly what the "device name" is: a human-usable representation of a VMS UCB.

Some have described this "unit-based" approach as the "geographic" or "topologic" approach. The Problem defined in this document is one of naming the slot where something plugs in, not the brick that plugs in to that slot.

1.2 Easy part vs. Hard part

One might further argue that the vast majority of cases entails disks permanently mounted clusterwide; and therefore the proper thing to do is come up with a solution to that part of the problem. Any other disks could be named in whatever hacked-up manner happens to work in any given release.

Here is the reasoning behind this argument. If you divide the problem into two cases: "always mounted" and "seldom used", then it is easy to solve the "always mounted" problem. Once you have guaranteed a valid volume exists on a unit, then naming and sharing is trivial to do -- you store some data in the home block and hack MOUNT if necessary. The "seldom used" disk units could then depart from standard usage since they tend not to play a key role in a cluster's operations. In particular, a "seldom used" disk could logically be assumed to be task-specific, unlikely to be shareable and if necessary could be given a variable-across-reboots device name.

I am somewhat wary of the above approach. It is a little bit like building a house on two separate foundations, which is inherently unwise. It misses the point that sharing is a function of topology -- cabling restricts how devices are shared. Cabling changes might thus invalidate sharing information hypothetically stored in a home block. Units may transition from one case to the other as customer needs change. It is not clear that splitting things into two cases reduces complexity or implementation effort, in the case of configurations that change over time. It would appear that a solid naming foundation, independent of higher layers, provides the best solution for dynamic environments. Furthermore since SCSI devices are highly portable it seems reasonable to assume that most environments will be "dynamic", meaning they will see occasional device configuration changes.

(This is a DECwrite artifact)

For the above reasons, we will *not* be looking to take the approach outlined in this section. All device units are equal, regardless of their intended or anticipated use.

None of this, however, absolves us from the responsibility to consider the anticipated usage of device names. <u>An enabling technology must "enable"</u>.

2 Naming: today, future

Currently there is a direct mapping from <device configuration> to <device name>. Specifically, a device name is constructed from:

"DK"	SCSI disk mnemonic ("MK" for tapes)
"c"	Controller letter, derived from port ID "PKc", itself derived from the point in time the port was located via autoconfigure. "Port" is the software analog of "bus adapter".
"ID*100"	Target ID times 100
+"lun"	plus LUN on the Target ID.
DKB504	Disk on 2nd SCSI Port (PKB0), target ID 5, LUN 4.

Most of the time the LUN is 0 but some controllers such as the HSZ permit creation of LUNs on targets. HSZ device names, by the way, are entirely "virtual" and bear no resemblance to how the hardware is configured on the HSZ. The HSZ is not unique in this respect. In any event, support for LUNs is required as we will be seeing them more often in the future.

2.1 Bus namespace expansion

There is a big problem on the way here. The number of target IDs is expanding to at least 16 and the number of LUNs to 64. After SCSI-3 we may see **32- or 64-bit wide** target IDs and LUNs, well beyond our capability to map in such a direct fashion.

2.2 Port configuration expansion

CPUs can have multiple ports, usually one port per bus. Under VMS this uses standard "controller letter" taxonomy, e.g., DKA, DKB... The contoller letter comes from the corresponding port letter (PKA, PKB...).

We will have to be able to handle more than 26 ports per system. Current TLaser systems can be configured with 192 ports, with an eventual capacity of 320 ports. While this is a theoretical capacity figure, unlikely to even be approached, it indicates a need to substantially expand the port namespace. Currently we claim to support only 18 ports per system, but this is primarily an artificial restriction based upon our testing capacity and of course the genuine problem of how to handle overflow after controller letter 'Z'.

2.3 How many initiators

Currently only 2 CPUs can share a bus but this is an implementation matter, not an architectural restriction. There can be 3 or more CPUs on a single shared bus, and we should design to that model. (GRYPHON is slated to offer support for 3 initiator CPUs).

2.4 Not just for SCSI anymore

While there is a pressing need for help specifically for SCSI in the 7.1 timeframe, the overall naming problem needs to be investigated independently of technology.

3 Restrictions on any proposed solution

An Investigation Report must describe a method of naming devices that satisfies the growth issues described in the previous section. In addition the report must address each of the following requirements. They are listed in rough priority order since it may be impossible to satisfy all restrictions simultaneously. The first 5, and the maximum name length, are "primary" or P0 restrictions.

3.1 Different device, different name

If allocation classes are in use, ambiguities can arise. If a cluster has nodes ABLE and BAKER, each with allocation class 47 and each with SCSI devices, then ABLE's device DKA100 and BAKER's device DKA100, different disks, would both have the "name" 47DKA100. We can't have this.

3.2 Same device, same name

In clusters with shared SCSI buses (one bus, controllers on each of two different nodes) the unit numbers will be the same but the controller designations may be different. This would happen if a bus were attached to port A on node ABLE and port B on node BAKER. The same disk might be 47DKA400 and 47DKB400. Various locking protocols require the controller letter to be consistent. This problem could be made to vanish by "configuration restrictions", which is basically what we do now. That is not a desirable option; we should minimize such restrictions since ultimately restrictions minimize what we can sell, opening the door to competitors.

3.3 Same name every time

Any device name must be invariant. Specifically (but not exclusively) the name must remain the same without regard to the order of booting of nodes in a cluster. The device name must persist across system shutdowns. However if a node name, address, device configuration or boot device changes then it is permitted to change device names for that node. Adding a new path to an existing device may not change the unit name.

3.4 Different media, same name

A device name must be independent of the media. In particular there might be no media in a slot at the time the slot needs to have a name. Removing one medium and inserting another may not cause a device name to change. (There may be situations where device jumpers can cause a TargetID or LUN to change, but this is considered an error).

An exception case arises where the device class changes, namely between disk and tape. If someone removes a disk from a slot and inserts a tape in the same slot, it is OK to change the device name from a disk-looking name to a tape-looking name, and vice-versa. Changing device types is not yet supported, but naming cannot prohibit this from being supported one day.

3.5 No configuration restrictions

There may be no common disk files anywhere on the cluster -- each node might boot from its own system disk. We cannot require the customer to supply shared storage, nor can we count on there being private nonshared storage. It must be possible to do failover. <u>This means a solution must not restrict itself to currently-known technology; it must be extensible.</u>

3.6 Compatibility issues

3.6.1 We do not wish to change VMS device naming syntax

There may be some slight changes, e.g., in order to handle more than 26 ports. But there is an upper limit of 15 characters that may not be exceeded.

3.6.2 Console

Console device names should match VMS names. We may not be able to provide 100% compatibility in this area, but should attempt to do as much as possible.

3.6.3 Layered products should not break

3.6.4 Any naming scheme must be compatible with Dollar

3.6.5 Tapes and disks should be easily distinguishable

3.6.6 Industry Trends

Any naming scheme must interoperate with future controllers that don't run VMS, for which VMS can't write code.

3.7 No human intervention

AUTOCONFIGURE must "auto configure". Human intervention should be eliminated or at least minimized to well-documented situations. This includes SYSGEN parameters.

3.8 Be helpful

Naming schemes should simplify work done at higher layers, as opposed to requiring higher layers to account for naming requirements. The new naming scheme should be no more difficult to comply with than the current scheme.

4 Deliverables

- A description of a method for addressing the needs described in section 3 above.
- A list of the modules that need to be added or changed and the rough extent of the change (minor, moderate, extensive).
- A first Investigation Report will be issued by mid-July, 1995. It will detail progress to date and define the course of future work.

Minutes by Krieger. "xx" indicates there is at least one change bar that addresses the issue.

xx 1) Problem statement is not clear

xx 2) Problem is solving geographic location

xx 3) Does this affect DSA, DR and DV device name ?

xx 4) Want generic device naming - NOT SCSI device naming... problem statement
Need focus on SCSI 7.1 but need to consider a long term solution

xx 5) need to consider MSCP, Shadowing, rms device id dependecies in investigation

xx 6) It is a waste of time - to investigate adapters and controllers... this causes considering technology dependent solutions...

xx Look at multiple technologies in the investigation

7) Investigate allocation contribution/dependencies to device naming problem...

xx 8) More precise to 3.3 -- new i/o path can not change device name

xx 9) DK/DK and DK/DU failover is not part of this problem - but should not preclude the failover problem

xx 10) clarification of 3.6.6 concerning controller widget dependencies.. remove storageworks and consider current scsi industry trends

xx 11) new scheme should be no more difficult than current scheme to comply with

12) Please consider CMD hawk product and Storage work products as possible device name dependencies...

xx 13) clarification on section 1.2 - which problem is being investigated

xx 14) must live inside "16" byte limit...

15) name format/syntax ? keep something similar to what we have... - need to consider the low paid operator who currently use some
of the device name to determine geographic slot...
- NEED TO CONSIDER SYSTEM MANAGERS...
- Can the device name be opaque - see nick Carr ?

xx 16) No new sysgen parameters ! (see richard critz - strong desired goal) Can not proprogate shirron's hack (sysgen parameter ...)