A Tutorial on SCSI-3 Persistent Group Reservations

(Version 1.1)

by Lee Duncan, SUSE Labs

December, 2012

Table of Contents

Introduction....1
Background....2
The Basics, at a High Level....3
How To Reserve a Disk For Dummies, Part 1: A Single Client....4
PROUT/REGISTER: Registering a Key with the Target....4
PRIN/READ-KEY: Reading Registered Keys from the Target....4
PROUT/RESERVE: Reserving the Target....5
PRIN/READ-RESERVATION: Reading Reservations from the Target....5
PROUT/RELEASE: Releasing Our Reservation....5
PROUT/REGISTER: Unregistering....5
Reservation Types In More Detail....7
In Table Form....7
In English....7
How To Reserve a Disk For Dummies, Part 2: Multiple Computers....9
Extenuating Circumstances....10
PROUT/CLEAR: Clearing All Registrations and Reservations....10
PRIN/REPORT-CAPABILTIES: 10
PROUT/REGISTER-AND-IGNORE: Re-registering....10
PROUT/REGISTER-AND-MOVE: Giving Away a Reservation....11
PROUT/PREEMPT: Preempting a Reservation....11
How the Different Group Reservation Types Really Work....12
WRITE EXCLUSIVE and EXCLUSIVE ACCESS Reservations....12
REGISTRANTS-ONLY Reservations....12
ALL REGISTRANTS Reservations....12
Advanced Topics....13
Scope: How Much of a Target Are We Talking About?....13
Persistence: What About After a Power Cycle Event?....13
Handling More Than One Target at a Time....13
Conclusions....14
Appendix: Reference Information....15

Introduction

Many have heard of SCSI-3 Persistent Group Reservations, under various names, such as Group Reservations, PGR, or even the less descriptive SCSI Reserve. In this document, I will call them PGR, or SCSI-3 Persistent Group Reservations.

This document assumes the reader has basic Disc and SCSI knowledge.

Background

SCSI-3 Persistent Group Reservations grew out of a need to allow groups of SCSI Initiators (computers) to work together on the ownership of a SCSI target (disc). The existing SCSI-2 commands, called RESERVE and RELEASE, were inadequate, since they were designed for one and only one initiator.

Examples in this tutorial are in the form of pseudo-code.

The Basics, at a High Level

The SCSI commands that support Persistent Group Reservations can be quite obtuse at first glance, so instead of referencing the SCSI specification, these commands will be described at a high level. A more detailed description will follow.

PGR Commands are about *reserving* a SCSI disc for one or more cooperating computers. Persistent Group Reservations are often used in computer *clusters*, as a way to coordinate cluster activities.

The PGR Commands are *cooperative*, in that reservations and their management can be shared by a group of cooperating computers. They are also *mandatory*, in that a PGR reservation is enforced even against computers that know nothing about the group.

There are two types of SCSI-3 PGR sub-commands: **PROUT** commands, which are reservation or registration commands being sent from the initiator (computer) to the target (disc), and **PRIN** commands, where the initiator is querying the target about reservations or registrations.

How To Reserve a Disk For Dummies, Part 1: A Single Client

Let's talk a little bit, first, about a simple case of a single initiator (computer) using PGR commands to reserve a target disc.

In this case, here's a high-level look at what has to be done:

1. The initiator **registers** a reservation *key* with the target.
2. The initiator **reserves** the target using its reservation *key* for either *Exclusive Access* (nobody else can read nor write to the device) or *Write Exclusive* (nobody else can write to the device).
3. The initiator uses target. The system enforces the reservation
4. The initiator **releases** the target reservation, using its *key*.
5. The initiator can **unregister** it's key. Note that this also **releases** the reservation, if present, so Step 4 could be skipped if this step is taken.

Here is a look at those steps in more detail.

PROUT/REGISTER: Registering a Key with the Target

The PGR commands use the concept of a key. Each and every PGR client must first register a unique key for a target disc, and then use that key when managing PGR reservations on that disc. Let's assume that we have never talked to this target disc before. To register a key with this target, we use the **PROUT *REGISTER*** command, passing in the current key (which is zero, by default), and our new reservation key, which we must remember, for future use:

```
PROUT/REGISTER(resvn-key=0,
               new-resvn-key=0x123abc)
```

This registers our computer (initiator) with the target device.

PRIN/READ-KEY: Reading Registered Keys from the Target

We can look to see if our reservation key is registered using the **PRIN *READ-KEYS*** command:

```
PRIN/READ-KEYS()

RESULT: 1 key:
    0x123abc
```

As you can see, this shows that there is one and only one key registered, and its value matches the key we just registered, as expected. We are the only registered user of this device.

PROUT/RESERVE: Reserving the Target

If we now want to reserve the disc, we need only supply our registered reservation key, using the **PROUT *RESERVE*** command:

```
PROUT/RESERVE(type=EXCLUSIVE ACCESS,
              resvn-key=0x123bc)
```

PRIN/READ-RESERVATION: Reading Reservations from the Target

Even though this returns **OK**, we may wish to verify that we have the exclusive reservation by asking the target about its reservations, using the **PRIN *READ-RESERVATION*** command:

```
PRIN/READ-RESVN()

RESULT: Key=0x123abc, type=EXCLUSIVE ACCESS
```

And, indeed, it shows that the target is reserved in “exclusive access” mode, as we requested, and our computer owns the reservation. I'll talk more on the types of reservations below.

PROUT/RELEASE: Releasing Our Reservation

Once we have had our way with the target, it is time to be a good citizen and release our reservation, using the **PROUT *RELEASE*** command, again passing in our registered reservation key, which must match the current reservation key:

```
PROUT/RELEASE(type=EXCLUSIVE,
              resvn-key=0x123abc)
```

If we read the reservation list again, we should see that there are no reservations.

PROUT/REGISTER: Unregistering

If you are watching close, you may ask why we do not remove our registration key as well, as long as we are cleaning up. The answer is that it does no harm to leave our key registered, particularly if we are going to use the device again. Never the less, we can unregister our computer if we wish, and that actually removes any reservation we might hold, so this step can be used in place of releasing the reservation, if desired. To unregister an initiator, simply re-register with the “null” key (which has value 0x0):

```
PROUT/REGISTER(resvn-key=0x123abc,
                    new-resvn-key=0)
```

After this step, a query of the target registration list would show there were no more registrants.

Reservation Types In More Detail

In Table Form

Now that we have the basic single-computer PGR usage model under our belt, let's talk about reservation types. In the previous section, “exclusive access” was picked as the reservation type, but there are several others, as shown in Table 1.

NAME	ACCESS Restrictions	Reservation Holder
WRITE EXCLUSIVE	Media-access write commands restricted to reservation holder	Single reservation holder
EXCLUSIVE ACCESS	Media-access commands restricted to reservation holder	Single reservation holder
WRITE EXCLUSIVE, REGISTRANTS ONLY	Media-access write commands restricted to registrants	Single reservation holder
EXCLUSIVE ACCESS, REGISTRANTS ONLY	Media-access commands restricted to registrants	Single reservation holder
WRITE EXCLUSIVE, ALL REGISTRANTS	Media-access write commands restricted to registrants	All registrants are reservation holders
EXCLUSIVE ACCESS, ALL REGISTRANTS	Media-access commands restricted to registrants	All registrants are reservation holders

Table 1: SCSI-3 Persistent Group Reservation Types

In English

Here's an explanation of these reservation types in sentence form:

Exclusive Access: don't let anybody else use my disc at all, and don't let anybody take my reservation.

Write Exclusive: don't let anybody else write to my disc, and don't let anybody take my reservation.

Exclusive Access, Registrants Only: Don't let anybody but my group of friends use the disc, and don't let anybody take my reservation.

Write Exclusive, Registrants Only: Don't let anybody but my group of friends write to the the disc, and don't let anybody take my reservation.

Exclusive Access, All Registrants: Don't let anybody but my group of friends use the disc, and share my reservation with all of them as well.

Write Exclusive, All Registrants: Don't let anybody but my group of friends write to the disc, and share my reservation with all of them as well.

You may wonder why all the talk about the reservation holder. It turns out the reservation goes away when the reservation holder goes away. If there are multiple reservation holders, then the reservation goes away when the last one unregisters.

In summary, Exclusive Access and Write Access limit access to a single reservation holder, and the reservation goes away when the holder unregisters. The Registrants Only pair of reservations limit access to the group of registrants, but the reservation still goes away when the single reservation holder unregisters. And lastly the All Registrants pair of reservations restrict access to the group of registrants, but the reservation only goes away when all registrants unregister.

How To Reserve a Disk For Dummies, Part 2: Multiple Computers

Now that we understand how a single computer can use PGR Commands to manage reservations on a target, it's time to look at a case involving multiple computers.

1. Each initiator **registers** a unique reservation *key* with the target. It is beyond the scope of this document to describe how they ensure uniqueness.
2. Multiple initiators may try to **reserve** the target using its reservation *key* for either *Exclusive Access* (nobody else can read nor write to the device) or *Write Exclusive* (nobody else can write to the device), but only one initiator will succeed.

3. The winning initiator uses target. The system enforces the reservation. Other initiators must coordinate with the reservation holder to access the target.
4. The reservation-holding initiator **releases** the target reservation, using its *key*.
5. Each initiator can **unregister** it's key. Note that this also **releases** the reservation, if present, when the reservation-holding initiator unregisters, so Step 4 could be skipped if this step is taken.

As you can see, these steps are very similar to those in the simpler case of a single computer using the target. In the real world, it is usually more complicated than this. For example, what if an initiator owns the reservation but then goes away, such as might happen if the operating system panic-ed. What about if a computer tries to register but finds some other unknown software has left an old registration for our computer on the device, and we don't know what the key might be. The next section describes the PGR commands designed to handle such cases.

Extenuating Circumstances

Since clients don't always play nice together, here are some of the ways PGR commands handle bad or unexpected behavior.

PROUT/CLEAR: Clearing All Registrations and Reservations

If a client wants to start from scratch, they need only use the **PROUT *CLEAR*** command to clear all registrants and any reservation information from the target device. The caller must supply a reservation key that it has registered.

PRIN/REPORT-CAPABILTIES: Checking for Optional Behavior

The client can get a report on the PGR capabilities of a target by supplying the **PRIN *REPORT-CAPABILITIES*** command. The target will report it's capabilities in the following areas:

- CRH: Compatible Reservation Handling
- SIP_C: Specify Initiator Ports Capable
- ATP_C: All Target Ports Capable
- PTPL_C: Persist Through Power Loss Capable
- TMV: Type Mask Valid
- Allow Commands

- PTPL_A: Persist Through Power Loss Active

Please see the T10 SCSI-3 Group Reservations specification for more details on these capabilities. (See **Appendix: Reference Information** for details)

PROUT/REGISTER-AND-IGNORE: Re-registering

If a client wants to register a key but the target already has a key registered for that client, the target will reject the request as a reservation conflict. This can happen, for example, if the client has died and been completely replaced. In such a case, the original key may not be known. This is where the **PROUT *REGISTER AND IGNORE*** command comes in handy, as it tells the target to replace the existing key for this client with the newly-supplied key.

PROUT/REGISTER-AND-MOVE: Giving Away a Reservation

If a client wishes, they can register a different client and give their reservation to them at the same time using the **PROUT *REGISTER AND MOVE*** command. After this operation, the newly-registered client has the reservation that the original client started out holding.

PROUT/PREEMPT: Preempting a Reservation

What happens if another client holds a reservation then goes away? In this case, the PGR standards allow for preemption: one of the registrants tells the target it wants to preempt the existing reservation, and then the caller becomes the new reservation holder. The **PROUT *PREEMPT*** command also has the ability to tell the target to abort all existing tasks during the preemption.

How the Different Group Reservation Types Really Work

Now that we've seen how a single client can reserve and release a SCSI target using PGR commands and how multiple clients can work together using PGR commands, let's look at how the different reservation types might work in the real world.

Let's assume there are multiple clients. Each client needs to register a unique key, then one of the clients needs to reserve the target. If multiple clients try to reserve the target at about the same time, only the first one will succeed, and subsequent attempts will get a RESERVATION CONFLICT. This can sometimes be used to somewhat randomly pick an *owner* of the target, and all other clients would become slaves to that owner.

WRITE EXCLUSIVE and EXCLUSIVE ACCESS Reservations

With these types of reservations, usually the goal is to let one and only one client own the disc. For example, in clustering code, this might be used to decide which client is in charge of the cluster. One client wins, and the other clients become slaves. In the event that the owner of the reservation dies or goes away without cleaning up, any one of the other registrants can preempt the reservation, thus taking ownership.

REGISTRANTS-ONLY Reservations

This is much like the previous case, except that all registrants can access the target media while the reservation is held. But, as in the previous case, if the reservation holder goes away one of the registrants has to take over by preempting the reservation.

ALL REGISTRANTS Reservations

Much like REGISTRANTS-ONLY, except that once the reservation is created, all registrants are treated as reservation holders. This means that if the original reservation creator goes away it does not matter, as long as there are other registrants.

Advanced Topics

There are a few advanced topics not covered here, but they should be mentioned. See the Appendix: Reference Information section for more information on these areas.

Scope: How Much of a Target Are We Talking About?

Those that created SCSI-3 Persistent Group Reservations seemed to plan for the day when one could reserve part of a target disk, commonly known as a **LUN**. Toward that end, many of the PGR Commands have a field for *scope*, but there is only one legal value for that field, and that value tells the PGR commands to reserve the *whole LUN*. Perhaps one day the standards will be updated to allow addressing portions of a LUN.

Persistence: What About After a Power Cycle Event?

The “P” in PGR stands for *Persistent*. This is because there was a need for targets to remember their reservation information across power-cycle events. This feature is not required, though, so a client must check the capabilities of the target before attempting to use this feature. See PRIN/REPORT-CAPABILTIES: Checking for Optional Behavior for information on how to check the target's capabilities.

Handling More Than One Target at a Time

The REGISTER and REGISTER-AND-IGNORE PGR Commands may support an option that tells the target to apply the registration to all target ports. Check the capabilities to see if the target supports this option (PRIN/REPORT-CAPABILTIES: Checking for Optional Behavior).

Conclusions

SCSI-3 Persistent Group Reservations are not mystical and should not be scary. Their function is quite logical, but the SCSI-3 standard obfuscates them. In this day and age of networked computers and storage, it is quite useful to be able to manage access to a disc, and the PGR Commands can help you accomplish this.

Appendix: Reference Information

1. The SCSI T10 Committee manages a standards documents for Persistent Group Reservations, called SPC-4 (SCSI Primary Commands, version 4). See http://t10.org, where you can usually find a draft version of this document for free. In particular, in revision 33 of this document, see sections “5.9 Reservations”, “6.13 PERSISTENT RESERVE IN Commands”, and “6.14 PERSISTENT RESERVE OUT Commands”.
2. The only known (so far) SCSI-3 PGR test suite requires an open-iscsi initiator, and uses the Python “nose” test suite, available at:

   git://github.com/gonzoleeman/open-iscsi-pgr-validate.git
3. If you use Linux, then you have probably heard of the sg3-utils package. On most platforms you can use platform-specific installation tools to install these utilities. For example, on SUSE SLE-11 or openSUSE, you can use “zypper in sg3_utils”. You can find project information at http://freecode.com/projects/sg3_utils. Several commands in this package are useful, but of particular interest is the “sg_persist” command, which can execute any of the PRIN or PROUT commands discussed in this document. The test suite mentioned above uses the commands in this package.