```
The OSD Standard
================
OSD (Object-Based Storage Device) is a T10 SCSI command set that is designed
to provide efficient operation of input/output logical units that manage the
allocation, placement, and accessing of variable-size data-storage containers,
called objects. Objects are intended to contain operating system and application
constructs. Each object has associated attributes attached to it, which are
integral part of the object and provide metadata about the object. The standard
defines some common obligatory attributes, but user attributes can be added as
needed.

See: http://www.t10.org/ftp/t10/drafts/osd2/ for the latest draft for OSD 2
or search the web for "OSD SCSI"

OSD in the Linux Kernel
=======================
osd-initiator:
  The main component of OSD in Kernel is the osd-initiator library. Its main
user is intended to be the pNFS-over-objects layout driver, which uses objects
as its back-end data storage. Other clients are the other osd parts listed
below.

osd-uld:
  This is a SCSI ULD that registers for OSD type devices and provides a testing
platform, both for the in-kernel initiator as well as connected targets. It
currently has no useful user-mode API, though it could have if need be.

exofs:
  Is an OSD based Linux file system. It uses the osd-initiator and osd-uld,
to export a usable file system for users.
See Documentation/filesystems/exofs.txt for more details

osd target:
  There are no current plans for an OSD target implementation in kernel. For all
needs, a user-mode target that is based on the scsi tgt target framework is
available from Ohio Supercomputer Center (OSC) at:
http://www.open-osd.org/bin/view/Main/OscOsdProject
There are several other target implementations. See http://open-osd.org for more
links.

Files and Folders
=================
This is the complete list of files included in this work:
include/scsi/
        osd_initiator.h   Main API for the initiator library
        osd_types.h       Common OSD types
        osd_sec.h         Security Manager API
        osd_protocol.h    Wire definitions of the OSD standard protocol
        osd_attributes.h  Wire definitions of OSD attributes

drivers/scsi/osd/
        osd_initiator.c   OSD-Initiator library implementation
        osd_uld.c         The OSD scsi ULD
        osd_ktest.{h,c}   In-kernel test suite (called by osd_uld)
        osd_debug.h       Some printk macros
        Makefile          For both in-tree and out-of-tree compilation
```

```
	Kconfig		Enables inclusion of the different pieces
	osd_test.c	User-mode application to call the kernel tests
```

The OSD-Initiator Library

osd_initiator is a low level implementation of an osd initiator encoder.
But even though, it should be intuitive and easy to use. Perhaps over time an
higher lever will form that automates some of the more common recipes.

init/fini:
- osd_dev_init() associates a scsi_device with an osd_dev structure
  and initializes some global pools. This should be done once per scsi_device
  (OSD LUN). The osd_dev structure is needed for calling osd_start_request().

- osd_dev_fini() cleans up before a osd_dev/scsi_device destruction.

OSD commands encoding, execution, and decoding of results:

struct osd_request's is used to iteratively encode an OSD command and carry
its state throughout execution. Each request goes through these stages:

a. osd_start_request() allocates the request.

b. Any of the osd_req_* methods is used to encode a request of the specified
   type.

c. osd_req_add_{get,set}_attr_* may be called to add get/set attributes to the
   CDB. "List" or "Page" mode can be used exclusively. The attribute-list API
   can be called multiple times on the same request. However, only one
   attribute-page can be read, as mandated by the OSD standard.

d. osd_finalize_request() computes offsets into the data-in and data-out buffers
   and signs the request using the provided capability key and integrity-
   check parameters.

e. osd_execute_request() may be called to execute the request via the block
   layer and wait for its completion. The request can be executed
   asynchronously by calling the block layer API directly.

f. After execution, osd_req_decode_sense() can be called to decode the request's
   sense information.

g. osd_req_decode_get_attr() may be called to retrieve osd_add_get_attr_list()
   values.

h. osd_end_request() must be called to deallocate the request and any resource
   associated with it. Note that osd_end_request cleans up the request at any
   stage and it must always be called after a successful osd_start_request().

osd_request's structure:

The OSD standard defines a complex structure of IO segments pointed to by
members in the CDB. Up to 3 segments can be deployed in the IN-Buffer and up to
4 in the OUT-Buffer. The ASCII illustration below depicts a secure-read with
associated get+set of attributes-lists. Other combinations very on the same
basic theme. From no-segments-used up to all-segments-used.

```
 ________OSD-CDB__________
|                         |
|read_len (offset=0)     -|---------\
|                         |         |
|get_attrs_list_length    |         |
|get_attrs_list_offset   -|----\    |
|                         |    |    |
|retrieved_attrs_alloc_len|    |    |
|retrieved_attrs_offset  -|----|----|-\
|                         |    |    | |
|set_attrs_list_length    |    |    | |
|set_attrs_list_offset   -|-\  |    | |
|                         | |  |    | |
|in_data_integ_offset    -|-|--|----|-|-\
|out_data_integ_offset   -|-|--|--\ | | |
\_________________________/ |  |  | | | |
                            |  |  | | | |
 _______OUT-BUFFER________  |  |  | | | |
|      Set attr list      |</  |  | | | |
|                         |    |  | | | |
|-------------------------|    |  | | | |
|   Get attr descriptors  |<---/  | | | |
|                         |       | | | |
|-------------------------|       | | | |
|    Out-data integrity   |<------/ | | |
|                         |         | | |
\_________________________/         | | |
                                    | | |
 ________IN-BUFFER________          | | |
|      In-Data read       |<--------/ | |
|                         |           | |
|-------------------------|           | |
|      Get attr list      |<----------/ |
|                         |             |
|-------------------------|             |
|    In-data integrity    |<------------/
|                         |
\_________________________/
```

A block device request can carry bidirectional payload by means of associating
a bidi_read request with a main write-request. Each in/out request is described
by a chain of BIOs associated with each request.
The CDB is of a SCSI VARLEN CDB format, as described by OSD standard.
The OSD standard also mandates alignment restrictions at start of each segment.

In the code, in struct osd_request, there are two _osd_io_info structures to
describe the IN/OUT buffers above, two BIOs for the data payload and up to five
_osd_req_data_segment structures to hold the different segments allocation and
information.

Important: We have chosen to disregard the assumption that a BIO-chain (and
the resulting sg-list) describes a linear memory buffer. Meaning only first and
last scatter chain can be incomplete and all the middle chains are of PAGE_SIZE.
For us, a scatter-gather-list, as its name implies and as used by the Networking
layer, is to describe a vector of buffers that will be transferred to/from the

wire. It works very well with current iSCSI transport. iSCSI is currently the only deployed OSD transport. In the future we anticipate SAS and FC attached OSD devices as well.

The OSD Testing ULD

TODO: More user-mode control on tests.

Authors, Mailing list

Please communicate with us on any deployment of osd, whether using this code or not.

Any problems, questions, bug reports, lonely OSD nights, please email:
   OSD Dev List <osd-dev@open-osd.org>

More up-to-date information can be found on:
http://open-osd.org

Boaz Harrosh <bharrosh@panasas.com>
Benny Halevy <bhalevy@panasas.com>

References

Weber, R., "SCSI Object-Based Storage Device Commands",
T10/1355-D ANSI/INCITS 400-2004,
http://www.t10.org/ftp/t10/drafts/osd/osd-r10.pdf

Weber, R., "SCSI Object-Based Storage Device Commands -2 (OSD-2)"
T10/1729-D, Working Draft, rev. 3
http://www.t10.org/ftp/t10/drafts/osd2/osd2r03.pdf