

# Sptlrpc Interoperability DLD

Eric Mei

July 2008

## 1 Functional Specification

### 1.1 Changes in NEW.0 (HEAD)

After MGS node is upgrade from OLD.x to NEW.0, system administrator should issue following command to convert the configuration logs to the new format:

```
mgs> lctl < EOF
device <MGS_name>
upgrade_conf_logs <fsname>
EOF
```

Before this conversion be executed, Lustre can still operate normally, but not be able to set security flavors. Only after the conversion we get fully functional sptlrpc.

The conversion could be executed at any time when NEW.0 MGS is running. It's supposed to be executed only once. But doing it multiple times will have no negative impact.

### 1.2 Changes in OLD.x (b1\_8)

OBD device configuration log interpreter should be modified to recognize LCFG\_SPTLRPC\_CONF record, and do a simple check.

## 2 Use Cases

### 2.1 upgrade

1. System upgrade from OLD.x to NEW.0;
2. Sysadmin issue *lctl upgrade\_conf\_logs* to upgrade configuration logs;
3. Sysadmin is able to set sptlrpc flavors;

## 2.2 downgrade

1. All sptlrpc rules have been removed from NEW.0 MGS;
2. System downgrade from NEW.0 to OLD.x;
3. Further downgrade from OLD.x to OLD, client OBD will print out warning about unknown command SPTLRPC\_CONF, but ignore the error. Operation continue normally.

## 3 Logical Specification

### 3.1 Upgrade Logs

This functionality should be added on NEW.0 (HEAD).

- Main ioctl handler.

```
int mgs_upgrade_conf_logs(*obd, *fsname) {
    /*
     * according to @fsname, find all existing logs:
     * <fsname>-client: client log
     * <fsname>-OSTnnnn: OST logs
     * <fsname>-MDTnnnn: MDT logs
     */
    log_list = find_logs(fsname);

    /* upgrade each log */
    for_each_log(logname, log_list) {
        do_upgrade_log(obd, fsname, logname);
    }
}
```

- upgrade one log.

```
int do_upgrade_log(*obd, *fsname, *logname) {
    /* create a empty temporary log */
    llog_create(tmp_logname);
    /* iterate through the log, write to a temporary log */
    llog_process(logname, upgrade_log_handler, tmp_logname);

    /* rename the temporary log to usually name */
    rename(tmp_logname, logname);
}
```

- log iteration handler.

```

/*
 * copy all non-obsolete records, and insert SPTLRPC_CONF
 * record right after SETUP of affected OBD.
 */
int upgrade_log_handler(*rec, *tmp_logname) {
    /*
     * skip all the records which are marked dead. It means
     * the blocks begins with LCFG_MARKER and have CM_SKIP set,
     * and end with LCFG_MARKER.
     */
    if (rec_inside_dead_marker(rec))
        return 0;

    /*
     * for device logs of MDT, OST, MDC, OSC, find the record
     * sequence of (0) -> ATTACH(1) -> SETUP(2) -> <RNEXT>(3),
     * SPTLRPC_CONF should be right after SETUP.
     */
    if (in_status_3(rec)) {
        if (<RNEXT> is SPTLRPC_CONF) {
            /* we already have SPTLRPC_CONF record, it means we're
             * upgrading an already upgraded log. do nothing here
             */
        } else {
            /* insert the SPTLRPC_CONF log */
            record_a_new_sptlrpc_conf(tmp_logname);
        }
        reset_status(0);
    }

    /* copy this record */
    record_lcfg(rec, tmp_logname);
    return 0;
}

```

## 3.2 Log Interpreter

This functionality should be added on OLD.x.

```

int class_process_config(*lcfg) {
    switch (lcfg->lcfg_command) {
        ...
    }
}

```

```
case LCFG_SPTLRPC_CONF:
    log = sptlrpc_conf_log_extract(lcfg);
    if (log->scl_nrule != 0) {
        CWARN("nrule is not 0, you should delete all “
            ”existing sptlrpc rules on MGS\n”);
    }

    /* in any case, ignore it and return success */
    return 0;
    ...
}
}
```

The *LCFG\_SPTLRPC\_CONF* is defined without *LCFG\_REQUIRED* bit, which means further downgrade to OLD the command will be ignored, only some warning will be print out about unknown command.

## 4 State Specification

### 4.1 Locking

Other configure operation on MGS will obtain in-memory *fs\_db* within *mgs->mgs\_sem*, then do the real operation within *fs\_db->fsdb\_sem*. So the upgrade procedure should be protected by *mgs->mgs\_sem* and/or *fs\_db->fsdb\_sem*:

```
upgrade_log(*obd, *fsname) {
    mutex_down(mgs->mgs_sem);
    fs_db = find_fs_db(fsname);
    if (fs_db)
        mutex_down(fs_db->fsdb_sem);

    do_upgrade()

    if (fs_db)
        mutex_up(fs_db->fsdb_sem);
    mutex_up(mgs->mgs_sem);
}
```

There's no need to refresh config DLM lock, because the upgrade will have zero effect on behavior of OBD devices.

## 4.2 Recovery

If MGS crashed during upgrade, there's be a temporary log file leave on disk, which will do no harm. It will be cleaned up when next time doing the upgrade or downgrade, or it could be removed manually by mount MGS device as ldiskfs.

## 5 Environment

Already detailed discussed in HLD.