

Long-term suitability of the LU-20808 MDT-to-MDT read-intent wire structs (change 69254)

Date: 2026-10-02. Scope: protocol and layout review of the wire structs in change 69254 against existing Lustre RPC formats (`lustre_idl.h`, `lustre_req_layout.h`, `layout.c`), with the connect flag from 69320 and the WBC series as context. This is not a code review of the implementation. 69254 is the first patch of a series that declares the structs; the handler, flag advertisement and client side come later, so missing implementation is expected and is not reported. The `OBD_CONNECT2` flag value and placement are out of scope (handled elsewhere); only the requirement to check the flag is covered.

1. Bottom line

The two new structs are conventional and cleanly laid out: fixed-width types, 8-byte aligned, no holes, explicit padding, swabbers, wirecheck/wiretest entries, and a want/valid bitmask handshake that lets old and new MDTs interoperate. The layout itself is not the main risk.

The risks are in the *contract around* the layout, which is hard to change once the structs ship:

1. **Error and validity semantics are underspecified** (section 4.1). Only the name lookup has a status. An item that is "not valid" can mean unread, too big, unsupported or failed, and the requester cannot tell which.
2. **One `mdrq_xattr_size` caps all six xattrs** (4.2), and an oversized value is silently dropped with no size hint.
3. **Extension goes through fixed reply slots** (4.3). The reply format already uses 10 of the 12 buffer slots the request capsule supports, and the fixed 40-byte structs cannot grow in place (their RMFs use exact-size checking), so the 16 bytes of padding in each is the only growth path.
4. **The "never sent by a client" rule is only a comment** (4.5). The `OBD_CONNECT_MDS_MDS` and `OBD_CONNECT2_DNE_INTENT` check must be made on the server for every request, because the reply exposes raw `trusted.*` xattrs and ACLs with no credential check.

5. **The patch and the design PDF describe different protocols** (4.9). The PDF has per-operation intents carrying credentials and child attributes, with parent-derived results; 69254 is a credential-free raw read. Which one the wire format must serve decides the request layout.

I recommend resolving 1, 2, 4 and 5 before landing and writing down the rules in 3 and 6; section 7 has the checklist. The design question in 5 (typed buffers versus an embedded OUT request) should be answered by the author explicitly, because it decides how this format grows.

2. What was read, and what could not be

Read in full: the diffs of 69254 (`e7f5efe`), 69320 (`6f929a8`), the WBC foundation patch LU-13047 (`7f07a0e` , change 36924), and the AGENTS.md patch LU-20860 (`977f862`), which adds `Documentation/internals.md` . I treated that as the 69351 internals document.

Limits you should know about:

- Gerrit and JIRA were unreachable from the review environment (egress policy). The LU-20808 comments were only available as second-hand summaries; I use them for context and do not rely on exact wording. The design PDF ("Reduce DNE Cross-MDT Metadata RPC", 19 pages) was later supplied by you and read first-hand; section 4.9 compares it with the patch.
- Only the foundation patch of the WBC series was available. I could not read the rest of that series, so section 6 covers what that patch shows plus public design material, and no more.
- Comparisons were made against two sources. The `lustre_idl.h` and `lustre_req_layout.h` you supplied are current (the former already contains the 69320 flag but not 69254); items checked there are marked "(supplied header)". For code those headers do not cover (`layout.c` , `out_handler.c` , `lprocfs_status.c`), I used a public mirror of `lustre-release` at `4a7bddaf` (author date 2024-11-01), which is older than the patches' base tree; those items are marked "(mirror)" and master may differ.

3. What 69254 defines

Item	Definition
Intent opcode	<code>IT_MDT_TO_MDT_READ = 0x00010000</code> , next free bit above <code>IT_BRW</code> (0x8000); retired 0x100, 0x200 and 0x2000 are not reused

Request	struct mdt_dne_read_req , 40 bytes: lu_fid mdrq_fid @0, __u32 mdrq_want @16, __u32 mdrq_xattr_size @20, two __u64 padding @24, @32
Reply	struct mdt_dne_read_rep , 40 bytes: __s32 mdrp_status @0, __u32 mdrp_valid @4, lu_fid mdrp_fid @8, two __u64 padding @24, @32
Item bits	MDR_FID 1 , MDR_ATTR 2 , MDR_ACL 4 , MDR_DEF_ACL 8 , MDR_PIN 0x10 , MDR_LMV 0x20 , MDR_LOV 0x40 , MDR_DEF_LMV 0x80 (the last six map to named xattrs)
Request format	RQF_LDLM_INTENT_DNE_READ , client: PTLRPC_BODY, DLM_REQ, LDLM_INTENT, DNE_READ_REQ, NAME
Reply format	server: PTLRPC_BODY, DLM_REP, DNE_READ_REP, DNE_READ_ATTR (fixed struct obdo), then six variable opaque buffers, one per xattr
Gating	peer must have agreed on OBD_CONNECT2_DNE_INTENT (defined separately in 69320)

The offsets and sizes in the patch's own wiretest entries match my arithmetic (no implicit padding; both structs are multiples of 8; `lu_fid` is 8-aligned at offsets 0 and 8).

What is good, for the record: explicit enum values (as the new AGENTS.md requires), no bitfields or unions, payload kept in separate RMF buffers like other intents rather than inside `ldlm_intent` , xattrs carried as opaque bytes identical to what `OUT_XATTR_GET` returns (so existing consumers already handle their byte order), no credentials on the wire, and a flag-only patch for the connect bit as AGENTS.md asks.

4. Findings

4.1 Validity and error contract (resolve before landing)

The reply says which items came back (`mdrp_valid`) and, for the name only, why resolution failed (`mdrp_status`). Gaps:

- **Three states collapse into two.** The patch text says a set bit with an empty buffer means "no such xattr"; an unset bit means the MDT did not read it. But "not read" covers unsupported item, value too big, read error and "lock did not cover it". The requester's only sane response is to fall back to `OUT_*` , which is fine, but it should be the *stated* rule, and the

wire should let it distinguish "too big" (retry with a bigger size) from "failed" (do not retry).

- **No per-item errno.** A failed attribute or xattr read has nowhere to go. The 16 bytes of reply padding could carry a small failure bitmask or one `__s32 mdrp_item_status`, but whatever is chosen has to be decided now.
- **mdrp_status versus the enqueue result.** For a create, `-ENOENT` on the name is the *success* case (a negative result that can be cached while the lock is held). The contract should say that `mdrp_status = -ENOENT` is a valid, cacheable reply, and how it differs from lock-not-granted or an RPC failure.
- **Name resolution in striped directories.** If the locked object is a striped master and the name hashes to another stripe, an `-ENOENT` here would be wrong. Patch 69260 keeps such lookups out of striped directories on the server, but the wire contract should define a status meaning "not authoritative here" so every implementation agrees and a requester never caches a false negative.
- **Redundant mdrq_fid.** The locked object is already identified by the lock resource in `DLM_REQ`. State which is authoritative and that a mismatch is an error.

4.2 Single xattr size cap

`mdrq_xattr_size` is one `__u32` applying to all six xattrs, and a value that does not fit "is not returned at all". Because the requester sets reply buffer sizes up front, one cap has to serve ACLs of a few hundred bytes and layouts that can reach tens of KB (the maximum stripe count is 2000 per the new internals.md). A cap sized for ACLs drops LOV and default LMV; a cap sized for layouts makes every reply buffer allocation large.

Options: per-item sizes (an array in the request, which fits in the 16 bytes of padding only for a few items), a total reply budget, or returning the needed size for dropped items so the fallback `OUT_XATTR_GET` can be right-sized. `OUT` already has `ou_result_size` per operation, which is the precedent for per-item sizing.

4.3 Extension model: slots, exact sizes and padding

- **Slot budget.** `REQ_MAX_FIELD_NR` is 12 (supplied header). The new reply format has 10 fields, so two spare slots remain. Each new readable item (linkEA and the encryption context were both listed as missing in the JIRA review) needs a new `RMF_*` and a new slot. The 24 spare bits in `mdrq_want` are

not the limiting resource; slots are. `REQ_MAX_FIELD_NR` is an in-memory array bound rather than a wire limit, so it can be raised, but that touches every `req_capsule`.

- **Exact-size RMFs.** Both structs use flags `0`, which means the received buffer must match `sizeof()` exactly (`RMF_F_NO_SIZE_CHECK` and `RMF_F_MINIMUM_SIZE` exist for the other cases). So a future version cannot simply make the struct larger: the only in-place growth is the padding. That is standard Lustre practice, but it should be a deliberate choice, and the padding rules in 4.4 become essential.
- **Appending buffers is the compatible path.** A newer MDT replying to an older requester adds trailing buffers the old format ignores; a newer requester with an older MDT finds trailing buffers absent and must test `mdrp_valid` first. Write that rule down and test it, since it is what makes new items possible without a new connect flag.
- **Alternative to six slots:** one variable "xattr bundle" buffer carrying name, length and value per entry, following the existing `RMF_EAVALS / RMF_EAVALS_LENS` precedent (mirror) used by the `getxattr` formats. That costs one slot permanently instead of one per xattr.

4.4 Reserved space and unknown bits

- Padding is never checked on receipt in typical Lustre code, so later reuse of a padding field is only safe if its meaning is announced by a bit. State: senders zero it; receivers ignore it; a field that takes over padding is announced by a new `mdrq_want / mdrp_valid` bit or a `flags` field.
- State that a responder ignores unknown `want` bits and never sets unknown `valid` bits. This is what lets new items ship without a connect flag each.
- Consider naming the padding `reserved` and adding a `flags` word now; two `__u64` per struct is generous but unnamed space invites ad hoc use.

4.5 Security: the gate must be enforced, not documented

The commit message says no credentials travel and the requester makes every permission decision. That is a sound trust model *between MDTs*, but then the reply to anyone who can send the opcode is raw `trusted.lov`, `trusted.lmv`, `trusted.dmv`, `trusted.pin` and both ACLs for an arbitrary FID. The IT bit lives in a shared `ldlm_intent` namespace that clients also use, so the "never sent by a client" comment is not a protection. The handler must reject the opcode unless the export has both `OBD_CONNECT_MDS_MDS` (already the established MDT-to-MDT gate in the target code, mirror) and

`OBD_CONNECT2_DNE_INTENT` , on every request rather than once at connect. The JIRA review raised this; it should also be a line in the wire contract.

4.6 Lock coverage and grant semantics

- A `valid` bit should be set only for items covered by the inodebits actually granted, since the requester will cache them for the life of the lock. The granted bits can be a subset of those requested. Put that in the contract.
- The JIRA review's `mdt_out` starvation concern ("execute the intent only if the lock is granted immediately") may need no new field: LDLM already has `LDLM_FL_BLOCK_NOWAIT` (mirror). Worth confirming it behaves correctly on this enqueue path before inventing a new request flag.
- Resend and replay need no wire field (the enqueue resend path covers it), but the contract should say the reply is recomputed under the already-granted lock.

4.7 Is a new format the right shape? (design question for the author)

The existing `OUT_UPDATE` already carries `OUT_ATTR_GET` , `OUT_XATTR_GET` , `OUT_INDEX_LOOKUP` and friends between MDTs, with per-operation results and sizes. Three ways to piggyback the same reads on the lock enqueue:

Option	Extensible	Per-item status and size	Cost
A. This patch: fixed struct plus typed buffers	New item = new bit, slot, RMF	No (4.1, 4.2)	Small, simple, closed read set
B. Embed an <code>object_update_request</code> of read-only ops in the enqueue	Any xattr by name, new ops for free	Yes (<code>our_rc</code> , <code>ou_result_size</code> already exist)	Larger headers; must whitelist read-only ops; runs <code>OUT</code> handlers in enqueue context
C. Reuse the <code>getattr-intent</code> reply buffers (<code>RMF_MDT_BODY</code> , <code>RMF_MDT_MD</code> , <code>RMF_DEFAULT_MDT_MD</code> , <code>RMF_ACL</code>)	Limited	Limited	Least new wire, but they are client-oriented forms and intent-specific

A is acceptable if the read set is genuinely closed. The JIRA review already lists two more items (`linkEA`, encryption context), which would take the reply

to 12 of 12 slots, so I would ask the author to state the intended final set and justify A over B in the commit message. The attribute carrier is consistent with the existing path: `out_attr_get` builds a `struct obdo` with `obdo_from_la()` (mirror), so A and B share a representation. That obdo is 208 bytes and, like `OUT_ATTR_GET`, has no field for birth time or rdev; confirm the OSP cache does not need them. Also confirm the fixed-size `RMF_DNE_READ_ATTR` tolerates a zero-length buffer when `MDR_ATTR` is not requested; I did not verify that `req_capsule` behavior.

4.9 Divergence between the design PDF and 69254

Read first-hand. The PDF and the patch describe different protocols, and the wire format should not be frozen until the author says which is intended.

Aspect	Design PDF	69254
Operations	Per-operation intents (create, rename, unlink prepare), entry mode vs object mode for rename	One read-only intent, <code>IT_MDT_TO_MDT_READ</code>
Request content	Proposed child attributes and the caller's credentials	FID, item mask, xattr size cap; no credentials
Reply content	Results derived by the parent MDT (permission, existence, etc.)	Raw <code>obdo</code> and opaque xattr blobs; the requester interprets them
Intent placement	Only on the name-hash enqueue	Attached to whichever enqueue the requester sends
Omissions	"Reply flags the omission"; three outcomes: incomplete preparation, advisory results, missing or truncated xattrs	<code>mdrp_valid</code> bits only; "advisory results" has no wire representation
Buffer set	<code>md_lock_intent</code> / <code>MLI_BUF_*</code> set, without the access ACL	Includes <code>MDR_ACL</code>

Consequences for the format:

- **Credentials.** If the PDF's model is intended, the request must carry uid/gid/groups/capabilities, which 4.5 currently treats as absent by design. A raw read that skips permission evaluation on the parent MDT shifts the check to the requester; that is a security decision, not just a layout one.

- **Opcode shape.** Per-operation opcodes and a read-only opcode do not share a request struct. Either 69254 is the first of a family (then the request needs an operation/mode field and a versioned tail, not two padding words), or it is the whole feature (then the PDF should be updated).
- **"Advisory results".** The third outcome has no field in the reply, so the requester cannot distinguish advisory from authoritative data. It must be specified, with `mdrp_status / mdrp_valid` semantics from 4.1.
- **Performance caveat.** The PDF reports a directory-stat regression of about 23%. That is the PDF's own measurement and I did not reproduce it; it matters here because piggybacked reads add reply size to every intent enqueue, which argues for the item mask staying request-driven.

4.8 Connect flag and naming

Per the requester, the value and placement of `OBD_CONNECT2_DNE_INTENT` in 69320 are handled elsewhere and are not assessed here. What matters for the structs is that the flag check is required (4.5): the handler for `IT_MDT_TO_MDT_READ` must refuse the intent on any export that has not negotiated it, since 69254 only declares the structs and the later patches implement the handler.

Naming is inconsistent across the series and becomes permanent uapi:

`IT_MDT_TO_MDT_READ` (69254), `IT_DNE_READ` (patch titles), `mdt_dne_read_*`, `MDR_*`, `RMF_DNE_READ_*`. Pick one family.

5. Process points from the new project guidance

The AGENTS.md and internals.md in 977f862 say new wire formats and `OBD_CONNECT_*` flags need agreement on lustre-devel and in Jira first, that connect flags land as flag-only patches, that wire structs get wirecheck entries with a regenerated `wiretest.c`, and that enum members have explicit values. 69254 and 69320 follow the pattern. One thing to check: the wiretest additions appear to be hand-edited into both copies (`lustre/ptlrpc/wiretest.c` and `lustre/utils/wiretest.c`); regenerate and diff against the generated output.

Also coordinate the IT bit `0x10000` with other in-flight series before landing, as the `lustre_idl.h` comment requires.

6. Fit with the Metadata Writeback Cache (WBC)

What the foundation patch (LU-13047, change 36924) shows on the wire:

- No new `RQF_*`, `RMF_*`, wire structs or connect flag. It adds two LDLM request flags, `LDLM_FL_INTENT_PARENT_LOCKED` (bit 6) and `LDLM_FL_INTENT_EXLOCK_UPDATE` (bit 7), and asks for an EX lock through the existing create intent (`IT_CREAT`). Flushing in this "lazy" mode uses ordinary RPCs.
- User-visible additions are `struct lu_wbc_state`, `LL_IOC_WBC_STATE`, and flush and cache mode enums. This is user ABI, not RPC protocol.
- (Mirror note: bit 6 is `LDLM_FL_CONTENTION` in my older mirror and unused elsewhere there; the WBC base tree evidently removed it. Worth a check that no older peer sets it.)

Consequences for 69254, with the caveat that the batching and flush patches of the series were not available to me:

1. **No collisions seen** in the IT bits, CONNECT2 bits or LDLM flag bits between this series and 69254/69320. Re-check when the later WBC patches are visible.
2. **Shared vocabulary.** A WBC client holding an EX subtree lock will need the same information the MDR items describe (attributes and inheritable defaults of a directory: default layouts, default ACL) to create children locally. 69254 is peer-only, so WBC cannot reuse it as written. If both uses are expected, keep the `MDR_*` items and reply independent of the opcode name and treat "peer only" as a gate, so a client-facing intent is an additive change rather than a second format.
3. **Batch containers.** The existing batch RPC embeds complete `lustre_msg` sub-requests (mirror), so formats defined as `RQF_*` compose into it, while flexible-array `OUT` payloads live in their own container. Option A is neutral to the WBC flush path; this matters only if the flush is built on the batch RPC.
4. **Server-side value remains.** Cross-MDT operations flushed from a WBC cache still go through a coordinator MDT, so the preparation reads this intent removes still occur on that path.

7. Recommendations, in order

1. Specify, in the `lustre_idl.h` comments, the contract in 4.1, 4.4 and 4.6: per-item meaning of an unset valid bit, status for "not authoritative", cacheability of `-ENOENT`, zero-padding, unknown-bit rules, and coverage by

granted inodebits.

2. Decide the size model now (4.2): per-item sizes or a budget plus a "needed size" hint for dropped items.
3. Require the `OBD_CONNECT_MDS_MDS` and `OBD_CONNECT2_DNE_INTENT` check per request in the handler (later patch in the series), and state it in the wire contract.
4. Have the author state the final intended read set and justify A over B (4.7) in the commit message; if the set will grow past two more items, plan the bundle buffer or the OUT embedding before the slots run out.
5. Name the padding `reserved`, add a `flags` word, and decide whether the fixed RMFs should stay exact-size.
6. Reconcile the request layout with the design PDF (4.9) before freezing it.
7. Settle one name family for the intent, structs and RMFs.
8. Regenerate wiretest and clear the IT bit with other in-flight series.

8. Questions for the author

- Is the read set closed? What happens to linkEA and the encryption context?
- What exactly does an unset `valid` bit mean to the requester, and which fallbacks does it trigger?
- Why typed buffers rather than a read-only `OUT` request in the enqueue?
- Will a client-facing variant ever be wanted (WBC)?
- Which inodebits must the granted lock cover for each item?
- Was `LDLM_FL_BLOCK_NOWAIT` considered for grant-immediately behavior?
- Is 69254 the first step of the PDF's per-operation design (credentials, child attributes, derived results) or a separate raw-read mechanism? See 4.9.
- Why does the PDF's `MLI_BUF_*` set omit the access ACL that the patch carries?

9. Verification notes

Checked against the supplied headers: `REQ_MAX_FIELD_NR` is 12;

`RMF_EAVALS`, `RMF_EAVALS_LEN`, `RMF_ACL`, `RMF_DEFAULT_MDT_MD` and the batch formats (`RQF_MDS_BATCH`, `RMF_BUT_*`) exist; `IT_*` bit 0x10000 is unused and 0x100, 0x200 and 0x2000 are retired; the `MDR_*` and `mdt_dne_read_*` names do not collide with existing symbols in either header; `struct obdo` is unchanged; `XATTR_NAME_PIN` and `XATTR_NAME_DEFAULT_LMV` exist.

Checked against the mirror (4a7bddaf): the largest existing format has 12

fields (the new reply has 10); RMF flag

semantics (RMF_F_NO_SIZE_CHECK , RMF_F_MINIMUM_SIZE , default exact match);

out_attr_get replies with an obdo from obdo_from_la() ; sizeof(struct obdo) is 208

per wiretest; XATTR_NAME_PIN , XATTR_NAME_LMV and

XATTR_NAME_DEFAULT_LMV exist; LDLM_FL_BLOCK_NOWAIT exists. Not verified:

req_capsule handling of zero-length fixed buffers, whether master differs

from the mirror, the full WBC series,

and JIRA comments beyond second-hand summaries. The design PDF was read

first-hand; table entries in 4.9 are from my reading of it and should be

checked against the author's intent.

Sources: Gerrit changes 69254, 69320, 69351 and 36924 (patch files provided

by the requester); LU-20808 comments (summaries only) and its design PDF (provided by the requester);

public WBC design slides (LUG 2023, LAD 2021).