SGI Multi-Rail Unit Test Plan

- Overview
- Unit Test Plan
  - Local Network Configuration
    - In-Range UT
    - Out-of-Range UT
    - Error UT
  - Remote Peer Configuration
    - Expected Behavior
    - In-Range UT
    - Out-of-Range UT
    - Error UT
  - Policy Configuration
    - In-Range UT
    - Error UT
  - General Configuration
  - Functional Requirements
    - Interface Selection and Message Sending Requirements
    - Dynamic NID Discovery
    - Debugging Requirements
    - Network interface Health
    - Backwards Compatibility Requirements
  - Performance Requirements
  - Misc Error Scenarios

Overview

The Unit Test Plan (UTP) will follow the same section breakdown as the Requirements in the Scope & Requirement Document.

The following types of tests shall be included where it makes sense.

1. In-Range UT - these are the test cases which cover normal operations.
2. Out-of-Range UT - these are the test cases which cover out of range scenarios:
    a. border cases
    b. race conditions
    c. unexpected events
        i. EX: Tearing down an active Network Interface
3. Error UT
    a. Error parameters
    b. Error Conditions
        i. network goes down unexpectedly
        ii. Wire gets disconnected, etc

Performance Testing cases will be a separate section in this document.

Unit Test Plan

Configuration tests should be done through the DLC direct interface, as well as the YAML interface.

Local Network Configuration

In-Range UT

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description
cfg-020	cfg-005, cfg-010, cfg-015,cfg-045, cfg-055, cfg-060, cfg-065	UT-0005	• Configure 3 NIDs on the same TCP network. • Show the NIDs
		UT-0010	• Configure 3 NIDs on the same IB network • Show the NIDs

		UT-0015	• Configure 3 NIDs on the same TCP/IB Network • Show the NIDs • Delete 1 NID from the TCP/IB Network • Show the NIDs

		UT-0020	• Configure 2 NIDs on tcp0/o2ib0
			• Configure 2 NIDs on tcp1/o2ib1
			• Show the NIDs
			• Delete 1st NID from tcp0
			• Delete 2nd NID from tcp0
			• Show NIDs
			  ◦ No more tcp 0 should exist
			  ◦ o2ib0 should be unaffected

cfg-025	cfg-005, cfg-010, cfg-015,cfg-045, cfg-055, cfg-060, cfg-065	UT-0025	• Configure the system to have 4 CPTs
			  ◦ options libcfs cpu_npartitions=4 cpu_pattern="0[0] 1[1] 2[2] 3[3]"
			• Configure 2 NIDs on tcp0
			  ◦ NID 1 should be on CPTs 0, 3
			  ◦ NID 2 should be on CPTs 1, 2
			• Show NIDs
			  ◦ proper CPT association should be displayed

cfg-035	cfg-040, cfg-045, cfg-055, cfg-060, cfg-065	UT-0030	• Configure the system to have 4 CPTs
			  ◦ options libcfs cpu_npartitions=4 cpu_pattern="0[0] 1[1] 2[2] 3[3]"
			• Configure 3 NIDs on tcp0
			  ◦ NID 1 should be on CPTs 0, 3
			  ◦ NID 2 should be on CPTs 1, 2
			  ◦ NID 3 should be on all CPTs
			• Show NIDs
			  ◦ proper CPT association should be displayed
			  ◦ NID 3 should exist on all CPTs

		UT-0035	• Configure 1st NID on tcp0 using the legacy ip2nets parameter from DLC • Show NIDs
		UT-0040	• Configure 1st NID on tcp*/o2ib* in the following ip2nets form: ◦ tcp(<eth intf>)[<cpt>] <pattern> • Show NIDs to ensure that the interface has been added to the correct CPTs

		UT-0045	• Configure 1st NID on tcp*/o2ib* in the following ip2nets form:
			  ◦ tcp(<eth intf>, <eth intf>, ...)[<cpt>] <pattern>
			  ◦ [<cpt>] can have only one value
			• Show NIDs to ensure that the interface has been added to the correct CPTs

		UT-0050	• Configure 1st NID on tcp*/o2ib* in the following ip2nets form: ◦ tcp(<eth intf>[<cpt>], <eth intf>[<cpt>], ...) <pattern> • Show NIDs to ensure that the interface has been added to the correct CPTs
cfg-060	cfg-065	UT-0055	Go through the following lnetctl commands and excercise their parameters: • net • set num_range

Out-of-Range UT

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description

cfg-020	cfg-005, cfg-010, cfg-015	UT-0060	• Configure 32 NIDs on the same TCP/IB Network • Show the NIDs
		UT-0065	• Configure 32 NIDs on the same TCP/IB Network • Show the NIDs. • Delete 32 NIDs on the same TCP/IB Network
		UT-0070	• Configure NID A, B and C on tcp0/o2ib0 Network • Configure NID A and B on tcp1/o2ib1 Network • Show the NIDs ◦ Configuration should succeed. NIs can exist on different networks
cfg-060	cfg-065	UT-0075	Go through the following lnetctl commands and excercise their parameters, by providing out of range values: • net • set num_range
		UT-0080	• Don't configure any LNet modprobe. • Load LNet where there exists only one commissioned IB interface with IPoIB configured • a TCP network should be created with that IB interface • Configure an o2ib network with the same IB interface • Now you should have two interfaces with exactly the same IB

Error UT

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description
		UT-0090	• Configure a non-existent NID on tcp0 • Configuration should fail with INVALID PARAMETER

		UT-0095	• Configure the system to have 4 CPTs
			  ◦ options libcfs cpu_npartitions=4 cpu_pattern="0[0] 1[1] 2[2] 3[3]"
			• Configure 3 NIDs on tcp0
			  ◦ NID 1 should be on CPTs 0, 4
			  ◦ NID 2 should be on CPTs 1, 2
			  ◦ NID 3 should be on all CPTs
			• Show NIDs
			  ◦ NID 1 should fail since no CPT 4

		UT-0096	• Configure 1st NID on tcp*/o2ib* in the following ip2nets form: ◦ tcp(<eth intf>, <eth intf>, ...)[<cpt, cpt>] <pattern> • Configuration should fail with syntax error

		UT-0100	Go through the following lnetctl commands and excercise their parameters, by providing error values:
			• net
			  ◦ Valid net values are: tcp, o2ib, gni
			  ◦ Provide any garbage. Return value should be BAD PARAM
			• set num_range
			  ◦ valid range is any positive value.
			  ◦ Provide a negative value. Return value should be BAD PARAM

		UT-0105	Delete a non-existent network Should return -EINVAL
		UT-0110	Delete a non existent NID on tcp/o2ib Should return -EINVAL

Remote Peer Configuration

Expected Behavior

- A peer can be added by specifying a list of NIDs
  - The first NID shall be used as the primary NID. The rest of the NIDs will be added under the primary NID
- A peer can be added by explicitly specifying the key NID, and then by adding a set of other NIDs, all done through one API call
- If a key NID already exists, but it's not an MR NI, then adding that Key NID from DLC shall convert that NI to an MR NI
- If a key NID already exists, and it is an MR NI, then re-adding the Key NID shall have no effect
- if a Key NID already exists as part of another peer, then adding that NID as part of another peer shall fail
- if a NID is being added to a peer NI and that NID is a non-MR, then that NID is moved under the peer and is made to be MR capable
- if a NID is being added to a peer and that NID is an MR NID and part of another peer, then the operation shall fail
- if a NID is being added to a peer and it is already part of that Peer then the operation is a no-op.

In-Range UT

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description
cfg-070		UT-0115	• add a new peer with only 1 NID
cfg-070		UT-0120	• add a new peer with only 1 NID • add more nids to that peer
cfg-070		UT-0125	• add a new peer with mulitple NIDs
cfg-070		UT-0130	• add a new peer with only 1 NID • Delete that NID
cfg-070		UT-0131	• add a new peer with multiple NIDs • delete the primary NI of the peer • The entire peer should be deleted.
cfg-070		UT-0135	• add a new peer with multiple NIDs • Delete each NID one at a time until the peer is removed
cfg-070		UT-0140	• add a new peer with multiple NIDs • Delete all NIDs but primary NID only. • Re-add multiple NIDs one at a time.
cfg-070		UT-0145	• add a new peer with multiple NIDs • Delete all NIDs but primaray NID. • Re-add multiple NIDs in one shot.
cfg-075		UT-0150	• add a new peer with multiple NIDs on different networks

Out-of-Range UT

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description
		UT-0155	• add a new peer with 32 NIDs
		UT-0160	• delete a peer and all its 32 NIDs
		UT-0165	• load lnet • lnetctl lnet configure • add 2 or more peers on a non-local network • delete peer 1 • delete peer 2

		UT-0170	• load lnet • lnetctl lnet configure • add 2 or more peers on a non-local network • lnetctl lnet unconfigure • lustre_rmmod
		UT-0171	• load lnet • lnetctl lnet configure • add 2 or more peers on tcp1 (non-local) • check that refcount = 2 (1 for hashlist & 1 for remote list) • check credits are not set • add tcp1 network • Check refcount 1 (remote list refcount removed) • check credits are set
		UT-0172	• load lnet • lnetctl lnet configure • add 2 or more peers on tcp1 (primary peer ni) • add 2 or more peers on tcp2 • add tcp 1 and tcp 2 networks • remove the tcp1 network • check that the entire peer is removed
		UT-0173	• same steps as above • remove a tcp2 network • check that all peers on that network are removed.
		UT-0175	• startup lnet • startup traffic • add a peer ni on a non-local network • add a local network for that peer • Send traffic over that peer_ni
		UT-0176	• startup lnet • add tcp1 network • add peers on tcp1 network • check they are multi-rail • run taffic • delete the peers • peers should be recreated because of traffic and they should be non-mr

Error UT

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description
cfg-070		UT-0180	• add more than 32 NIDs
cfg-070		UT-0185	• add a peer with multiple NIDs • delete a non-existent peer NID from the peer identified by key-NID
cfg-080	snd-065	UT-0190	• add peer 1 with NIDs A, B and C • add peer 2 with NIDs D, C and E ◦ Adding NID C should fail

Policy Configuration

In-Range UT

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description

cfg-090		UT-0195	• Set the NUMA range to 0 • The NI closest to the message memory NUMA will be picked.
cfg-090		UT-0200	• Increase the NUMA range step by step • Note that more NIs are picked when sending
cfg-090	snd-025	UT-0205	• Set the NUMA range to a large value • start traffic • NIs are picked in round robin

Error UT

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description
cfg-090		UT-0210	• Set the NUMA range to < 0 • This should be rejected

General Configuration

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description
cfg-170		UT-0215	• Configure multiple NIs • Configure multipe Peers with multiple NIDs • set NUMA range value • Dump the YAML configuration • use the YAML configuration file to delete all configuration • use the YAML configuration file to reconfigure the node.

Functional Requirements

Interface Selection and Message Sending Requirements

Note to test NUMA proximity, you can use python psutil to bind a process to a specific CPU then execute a write/read operation to the FS on that CPU.

The CPU distances can be acquired from /proc/sys/lnet/cpu_partition_distance

The NUMA cpu list can be acquired from /sys/devices/system/node/node*/cpulist

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description	Behavior of Note
snd-005		UT-0220	• Configure 3 NIs with equadistant NUMA distance • Send three or more messages • Dump statistics on each NI to verify that each NI was used to send messages	
snd-010	snd-015	UT-0225	• Configure 3 NIs closer to different NUMA nodes • dump the NI statistics • Verify that each NI has the correct device CPT	
snd-020		UT-0230	• Configure 3 NIs with different NUMA distances • Send messages • Confirm through statistics that messages are being sent over the nearest NI (NUMA wise)	

snd-020		UT-0235	• Configure 2 NIs with different NUMA distances • Send messages • Confirm through statistics that messages are being sent over the nearest NI (NUMA wise) • add another NI which is close NUMA wise than the current nearest • confirm through statistics that messages are not being sent over the newly added NI	
snd-030		UT-0240	• Configure 3 NIs, one EDR, one FDR and one QDR • set the NUMA range to a large value so all NIs are considered through RR • start traffic • monitor statistics on each NI. • Confirm that EDR is preferred until it becomes saturated, then FDR is selected then QDR	
snd-030		UT-0245	• Configure 3 NIs • set the NUMA range to a large value so all NIs are considered through RR • start traffic • monitor statistics on each NIs to confirm all are being used. • Remove one of the NIs • Confirm that that NI is no longer used for new messages • Confirm that the other 2 NIs are being used. • No messages should be dropped.	
snd-035		UT-0250	• Configure 3 NIs • Configure a peer with 3 NIDs • Send messages to the peer • Confirm through statistics that peer NIDs are being used based on their available credits.	
snd-040		UT-0255	• Configure 3 NIs which are not equadistant all on the same network • configure a peer with 3 NIDs all on the same network • start traffic • Confirm closest NUMA NI is being used • Confirm peer NIDs are being used • set NUMA range to a large value • Confirm all NIs are being used • Confirm no change in traffic pattern to the peers	
snd-045	snd-070	UT-0260	• Configure NIs A, B and C • Configure the peer with the same NIDs • Send 1 message which requires a response from NI A • Confirm that responses are being sent to the same NI	
snd-050		UT-0265	• Configure NIs A, B and C • Configure the peer with the same NIDs • Send 1 message which requires a response from NI A • bring down NI A • confirm that response is sent to one of the other configured NIDs	
snd-050		UT-0270	• Configure an MR system • Start traffic • monitor traffic is being sent to all configured peers • bring down one of the peer NIDs • monitory traffic is no longer sent to that peer NID • no messages should be dropped	

snd-050	snd-060, snd-075	UT-0275	• Configure an MR system
			• Start traffic
			• monitor traffic is being sent to all configured peers
			• bring down one of the peer NIDs
			• monitory traffic is no longer sent to that peer NID
			• bring up the peer NID again
			• monitor traffic is being sent to it again
				When a peer NID is removed then added or if a new peer NID is added, then it's peer_ni->lpni_seq number will start off at 0.
				
				In the case of credits being == then the newly added peer will always be picked until it's lpni_seq number catches up with the other peer NIs sequence numbers. See code below.
				
				`} else if (lpni->lpni_txcredits == best_lpni_credits) {`
				`/*`
				`* The best peer found so far and the current peer`
				`* have the same number of available credits let's`
				`* make sure to select between them using Round`
				`* Robin`
				`*/`
				`if (best_lpni) {`
				`if (best_lpni->lpni_seq <= lpni->lpni_seq)`
				`continue;`
				`}`
				
				This behavior will manifest itself in low bandwidth environment. In high bandwidth environment it is likely that the credits in the selection algorithm will be different and the peer_NI will be picked according to credits.
				
				Another scenario to consider is when the lpni_seq number wraps. In low bandwidth environment this could cause the peer NI which wrapped to be picked until it catches up with the other peer NIs sequence numbers.
				
				This in itself might not be significant enough, but does raise the question of the benefit of having a seq number to start with. Does it give much of a functional advantage, or having the credits criteria enough.
				
				The same issue is present with local NI sequence numbers.

snd-055	snd-060, snd-075	UT-0280	• Configure an MR system • Start traffic • monitor traffic is being sent to all configured peers • bring down one of the peer NIDs • monitory traffic is no longer sent to that peer NID • bring down all peer NIDs • message should fail.	
snd-055	snd-060, snd-075, snd-085	UT-0285	• Configure an MR system • Start traffic • monitor traffic is being sent to all configured peers over all NIs • bring down the local NIs one by one • note traffic is migrated to the NIs still up, until no NIs are left then messages are dropped	
snd-055	snd-060, snd-075, snd-085	UT-0290	• Configure an MR system • Start traffic • monitor traffic is being sent to all configured peers over all NIs • bring down the local NIs one by one • note traffic is migrated to the NIs still up, until no NIs are left then messages are dropped • bring up the NIs again and confirm that NIs are being reused.	
snd-055	snd-060, snd-075	UT-0295	• Configure two networks tcp and o2ib • Configure nodes to have multiple interfaces on each of the networks • start traffic over the o2ib network • o2ib should be used • bring down the o2ib network • traffic should migrage to the tcp network. • no traffic should be dropped.	

snd-080		UT-0300	• Configure an MR system • bring down an NI • confirm that the show info shows the NI as down	
snd-080		UT-0305	• TODO: how do we test device failure?	
		UT-0310	• Configure an MR system • Configure peers via DLC • Run traffic • Delete one of the peer_nis we're sending to via DLC • Traffic going over that peer_ni should continue but no more traffic should use that NI	
		UT-0315	• Configure an MR system • Configure peers via DLC • Run traffic • Delete one of the peer_nis we're sending to via DLC • Bring that peer_ni back • Note traffic stops and starts on that peer with no traffic loss • Repeat the deletion and reconfiguration of the peer_ni	
		UT-0320	• Configure an MR system • Configure peers via DLC • Run traffic • Delete the entire peer • The peer should be recreated on the next message, but it won't be MR capable.	

Dynamic NID Discovery

The unit tests for peer NID discovery depend on `lnetctl ping` not triggering discovery. To force discovery, use `lnetctl discover`. Note that some of the tests require DLC configuration to include non-existing peer NIDs. These nids are marked with a *.

Tests with discovery enabled.

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description
	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040	UT-DD-CFG-0001	• turn on lnet discovery • run lnetctl discover • ensure that peers are discovered
		UT-DD-CFG-0002	• turn off lnet discovery • run lnetctl discover • ensure we get some valid error back
		UT-DD-CFG-0003	• show the status of lnet discovery (if it's on or off)
		UT-DD-CFG-0004	• Make sure lnet discovery is configurable via YAML
		UT-DD-CFG-0005	• configure lnet_max_interfaces from command line • Show lnet_max_interfces from command line and ensure it's set
		UT-DD-CFG-0006	• configure lnet_max_interfaces from YAML • show lnet_max_interfaces from YAML and ensure it's set

	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040	UT-DD-EN-0001	Basic functionality 1-1: discovery of an MR peer via its primary. • MR Node with interfaces N1, N2 • MR Peer with interfaces P1, P2, P3 • Ping P1 from node • Ping P2 from node • Verify that node sees two different peers: P1, P2. • Discover P1 from node • Verify that node sees one MR peer with three NIDS: P1, P2, P3. • Verify that peer sees node as one MR peer with two NIDS: N1, N2.
	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040	UT-DD-EN-0002	Basic functionality 1-2: discovery of an MR peer via a secondary. • MR Node with interfaces N1, N2 • MR Peer with interfaces P1, P2, P3 • Ping P1 from node • Ping P2 from node • Verify that node sees two different peers: P1, P2. • Discover P2 from node • Verify that node sees one MR peer with three NIDS: P1, P2, P3. • Verify that peer sees node as one MR peer with two NIDS: N1, N2.
	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040	UT-DD-EN-0003	Basic functionality 1-3: discovery of an MR peer via a tertiary. • MR Node with interfaces N1, N2 • MR Peer with interfaces P1, P2, P3 • Ping P1 from node • Ping P2 from node • Verify that node sees two different peers: P1, P2. • Discover P3 from node • Verify that node sees one MR peer with three NIDS: P1, P2, P3. • Verify that peer sees node as one MR peer with two NIDS: N1, N2.
dyn-020	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040, dyn-055	UT-DD-EN-0004	Basic functionality 1-4: implicit discovery of an MR peer • MR Node with interfaces N1, N2 • MR Peer with interfaces P1, P2, P3 • Force some filesystem traffic between node and peer. • Verify that node sees one MR peer with three NIDS: P1, P2, P3. • Verify that peer sees node as one MR peer with two NIDS: N1, N2.
	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040	UT-DD-EN-0005	Basic functionality 1-5: discovery of an MR peer with > 16 interfaces. (This test exercises the code path that resizes the push buffers.) • MR Node with interfaces N1, N2, ..., N17 • MR Peer with interfaces P1, P2, ..., P17 • Ping P1 from node. • Discover P1 from node • Verify that node sees one MR peer with all NIDS: P1, P2, ..., P17 • Verify that peer sees node as one MR peer with all NIDS: N1, N2, ..., N17.
	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040	UT-DD-EN-0006	Compatibility 2-1: discovery of a non-MR peer via its primary. • MR Node with interface N1, N2. • Non-MR Peer with interfaces P1, P2, P3. • Ping P1 from node. • Ping P2 from node. • Verify that node sees two different peers: P1, P2. • Discover P1 from node • Verify that node sees one non-MR peer with three NIDS: P1, P2, P3. • Verify that peer sees one as one peer with one NID: N1.
	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040	UT-DD-EN-0007	Compatibility 2-2: discovery of a non-MR peer via a secondary. • MR Node with interface N1, N2. • Non-MR Peer with interfaces P1, P2, P3. • Ping P1 from node. • Ping P2 from node. • Verify that node sees two different peers: P1, P2. • Discover P2 from node • Verify that node sees one non-MR peer with three NIDS: P1, P2, P3. • Verify that peer sees one as one peer with one NID: N1.

	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040	UT-DD-EN-0008	Compatibility 2-3: discovery of a non-MR peer via a tertiary. • MR Node with interface N1, N2. • Non-MR Peer with interfaces P1, P2, P3. • Ping P1 from node. • Ping P2 from node. • Verify that node sees two different peers: P1, P2. • Discover P3 from node • Verify that node sees one non-MR peer with three NIDS: P1, P2, P3. • Verify that peer sees one as one peer with one NID: N1.
dyn-020	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040, dyn-055	UT-DD-EN-0009	Compatibility 2-4: implicit discovery of an MR peer • MR Node with interfaces N1, N2 • MR Peer with interfaces P1, P2, P3 • Force some filesystem traffic between node and peer. • Verify that node sees one non-MR peer with three NIDS: P1, P2, P3. • Verify that peer sees node as one peer with one NID: N1
dyn-060	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040	UT-DD-EN-0010	Interaction with DLC 3-1: DLC overrides Discovery of MR peer • MR node with interface N1 • MR peer with interface P1, P2, P3 • DLC configure MR peer on node with interfaces P1, P2, P4*. • Discover P1 from node. • Verify that node sees one MR peer with three NIDS: P1, P2, P4*. • Verify presence of error messages on node (error code is `-EPERM`): ◦ `Error adding NID P3 to peer P1: -1` ◦ `Error deleting NID P3 from peer P1: -1`
dyn-060	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040	UT-DD-EN-0011	Interaction with DLC 3-2: DLC overrides Discovery of non-MR peer • MR node with interface N1 • non-MR peer with interface P1, P2, P3 • DLC configure non-MR peer on node with interfaces P1, P2, P4*. • Discover P1 from node. • Verify that node sees one non-MR peer with three NIDS: P1, P2, P4*. • Verify presence of error messages on node (error code is `-EPERM`): ◦ `Error adding NID P3 to peer P1: -1` ◦ `Error deleting NID P3 from peer P1: -1`
dyn-060	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040, dyn-065	UT-DD-EN-0012	Interaction with DLC 3-3: DLC overrides Discovery of MR peer with primary conflict • MR node with interface N1 • MR peer with interface P1, P2, P3 • DLC configure MR peer on node with interfaces P2, P3, P4*. • Discover P2 from node. • Verify that node sees one MR peer with three NIDS: P2, P3, P4*. • Verify presence of error message on node (error code is `-EEXIST`): ◦ `Primary NID error P2 versus P1: -17`
dyn-060	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040, dyn-065	UT-DD-EN-0013	Interaction with DLC 3-4: DLC overrides Discovery of non-MR peer with primary conflict • MR node with interface N1 • non-MR peer with interface P1, P2, P3 • DLC configure non-MR peer on node with interfaces P2, P3, P4*. • Discover P2 from node. • Verify that node sees one MR peer with three NIDS: P2, P3, P4*. • Verify presence of error message on node (error code is `-EEXIST`): ◦ `Primary NID error P2 versus P1: -17`

dyn-060	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040	UT-DD-EN-0014	Interaction with DLC 3-5: "push MR bit" exception to DLC overrides Discovery
			• MR node with interface N1
			• MR peer with interface P1, P2, P3
			• DLC configure non-MR peer on node with interfaces P1, P2, P4*.
			• Discover N1 from peer.
			• Verify that node sees one MR peer with three NIDS: P1, P2, P4*.
			• Verify presence of error message on node (error code is `-EEXIST`):
			  ◦ `Push says P1 is Multi-Rail, DLC says not`
			  ◦ `Error adding NID P3 to peer P1: -1`
			  ◦ `Error deleting NID P3 from peer P1: -1`

dyn-060	dyn-005, dyn-015, dyn-025, dyn-030, dyn-035, dyn-040, dyn-065	UT-DD-EN-0015	Interaction with DLC 3-6: "push MR bit" exception to DLC overrides Discovery
			• MR node with interface N1
			• MR peer with interface P1, P2, P3
			• DLC configure non-MR peer on node with interfaces P2, P3, P4*.
			• Discover N1 from peer.
			• Verify that node sees one MR peer with three NIDS: P2, P3, P4*.
			• Verify presence of error message on node (error code is `-EEXIST`):
			  ◦ `Push says P2 is Multi-Rail, DLC says not`
			  ◦ `Primary NID error P2 versus P1: -17`

Tests with discovery disabled. Note that disabling discovery does not fully disable it. The MR capable node will continue to process pushes, and if there is a problem with a push it will ping the originator to obtain the information.

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description
	dyn-005, dyn-025, dyn-030	UT-DD-DIS-0001	Discovery disabled 4-1: discovery of an MR peer via its primary • MR Node with interfaces N1, N2 • MR Peer with interfaces P1, P2, P3 • Discovery is disabled on Node • Ping P1 from node • Ping P2 from node • Verify that node sees two different peers: P1, P2. • Discover P1 from node • Verify that node sees two different peers: P1, P2. • Verify that peer sees node as one peer with one NID: N1
	dyn-005, dyn-025, dyn-030	UT-DD-DIS-0002	Discovery disabled 4-2: discovery of an MR peer via a secondary • MR Node with interfaces N1, N2 • MR Peer with interfaces P1, P2, P3 • Discovery is disabled on Node • Ping P1 from node • Ping P2 from node • Verify that node sees two different peers: P1, P2. • Discover P2 from node • Verify that node sees two different peers: P1, P2. • Verify that peer sees node as one peer with one NID: N1
	dyn-005, dyn-025, dyn-030	UT-DD-DIS-0003	Discovery disabled 4-3: discovery of an MR peer via a tertiary. • MR Node with interfaces N1, N2 • MR Peer with interfaces P1, P2, P3 • Discovery is disabled on Node • Ping P1 from node • Ping P2 from node • Verify that node sees two different peers: P1, P2. • Discover P3 from node • Verify that node sees three different peers: P1, P2, P3. • Verify that peer sees node as one peer with one NID: N1
	dyn-005, dyn-025, dyn-030	UT-DD-DIS-0004	Discovery disabled 4-4: implicit discovery of an MR peer • MR Node with interfaces N1, N2 • MR Peer with interfaces P1, P2, P3 • Discovery is disabled on both Node and Peer • Force some filesystem traffic between node and peer. • Verify that node sees one peer with one NID: P1. • Verify that peer sees node as one peer with one NID: N1.

dyn-020	dyn-005, dyn-025, dyn-030, dyn-055	UT-DD-DIS-0005	Discovery disabled 4-5: implicit discovery of an MR peer. (This test shows that if discovery is enabled on either node or peer, it happens on both.) • MR Node with interfaces N1, N2 • MR Peer with interfaces P1, P2, P3 • Discovery is disabled on Node • Force some filesystem traffic between node and peer. • Verify that node sees one MR peer with NIDs: P1, P2, P3 • Verify that peer sees node as one peer with NIDs: N1, N2.
dyn-020	dyn-005, dyn-025, dyn-030, dyn-055	UT-DD-DIS-0006	Discovery disabled 4-6: implicit discovery of an MR peer, > 16 interfaces. (This test shows that if discovery is enabled on either node or peer, it happens on both, including retries required because buffers need to be extended.) • MR Node with interfaces N1, N2, ..., N17 • MR Peer with interfaces P1, P2, ..., P17 • Discovery is disabled on Node • Force some filesystem traffic between node and peer. • Verify that node sees one MR peer with all NIDs: P1, P2, ..., P17 • Verify that peer sees node as one MR peer with all NIDs: N1, N2. ... , N17.
dyn-060	dyn-005, dyn-025, dyn-030	UT-DD-DIS-0007	Disabled with DLC 5-1: DLC overrides Discovery of MR peer • MR node with interface N1 • MR peer with interface P1, P2, P3 • Discovery is disabled on node. • DLC configure MR peer on node with interfaces P1, P2, P4*. • Discover P1 from node. • Verify that node sees one MR peer with three NIDs: P1, P2, P4*. • No error messages should show up.
dyn-060	dyn-005, dyn-025, dyn-030	UT-DD-DIS-0008	Disabled with DLC 5-2: DLC overrides Discovery of non-MR peer • MR node with interface N1 • non-MR peer with interface P1, P2, P3 • Discovery is disabled on node. • DLC configure non-MR peer on node with interfaces P1, P2, P4*. • Discover P1 from node. • Verify that node sees one non-MR peer with three NIDS: P1, P2, P4*. • No error messages should show up.
dyn-060	dyn-005, dyn-025, dyn-030	UT-DD-DIS-0009	Disabled with DLC 5-3: DLC overrides Discovery of MR peer with primary conflict • MR node with interface N1 • MR peer with interface P1, P2, P3 • Discovery is disabled on node. • DLC configure MR peer on node with interfaces P2, P3, P4*. • Discover P2 from node. • Verify that node sees one MR peer with three NIDs: P2, P3, P4*. • No error messages should show up.
dyn-060	dyn-005, dyn-025, dyn-030	UT-DD-DIS-0010	Disabled with DLC 5-4: DLC overrides Discovery of non-MR peer with primary conflict • MR node with interface N1 • non-MR peer with interface P1, P2, P3 • Discovery is disabled on node. • DLC configure non-MR peer on node with interfaces P2, P3, P4*. • Discover P2 from node. • Verify that node sees one MR peer with three NIDS: P2, P3, P4*. • No error messages should show up.

dyn-060	dyn-005, dyn-025, dyn-030	UT-DD-DIS-0011	Disabled with DLC 5-5: "push MR bit" exception to DLC overrides Discovery
			
			• MR node with interface N1
			• MR peer with interface P1, P2, P3
			• Discovery is disabled on node.
			• DLC configure non-MR peer on node with interfaces P1, P2, P4*.
			• Discover N1 from peer.
			• Verify that node sees one MR peer with three NIDS: P1, P2, P4*.
			• Verify presence of error message on node (error code is `-EPERM`):
			  ◦ `Push says P1 is Multi-Rail, DLC says not`
			  ◦ `Error adding NID P3 to peer P1: -1`
			  ◦ `Error deleting NID P3 from peer P1: -1`

dyn-060	dyn-005, dyn-025, dyn-030, dyn-065	UT-DD-DIS-0012	Disabled with DLC 5-6: "push MR bit" exception to DLC overrides Discovery
			• MR node with interface N1
			• MR peer with interface P1, P2, P3
			• Discovery is disabled on node.
			• DLC configure non-MR peer on node with interfaces P2, P3, P4*.
			• Discover N1 from peer.
			• Verify that node sees one MR peer with three NIDS: P2, P3, P4*.
			• Verify presence of error message on node (error code is `-EEXIST`):
			  ◦ `Push says P2 is Multi-Rail, DLC says not`
			  ◦ `Primary NID error P2 versus P1: -17`

Debugging Requirements

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description

dbg-005	dbg-010, dbg-015, dbg-020, dbg-025, dbg-030, dbg-035, dbg-080	UT-0325	• dump per NI statistics
			  ◦ transmitted
			  ◦ received
			  ◦ dropped
			  ◦ timeouts
			  ◦ state

dbg-040	dbg-080, dbg-095	UT-0330	• configure multiple NIs • run traffic • dump stats on all NIs
dbg-040	dbg-080	UT-0335	• configure multiple NIs • run traffic • dump stats on all NIs • Filter on specific NID
dbg-045	dbg-080	UT-0340	• dump LNet level statistics
dbg-050	dbg-080, dbg-100	UT-0345	• configure multiple peers • start traffic • dump per peer statistics
dbg-110		UT-0350	• configure multiple NIs • toggle their state from ACTIVE to DOWN • confirm that state change is being printed to console.
dbg-115		UT-0355	• configure an MR system • start traffic • bring down an NI • confirm that messages indicating that another NI/peer is being used is printed.
dbg-120		UT-0360	• Configure an MR system • run traffic • stop traffic • dump NI statistics • dump peer statistics • dump LNet level statistics • zero out stats • dump all statistics above to confirm they've been zeroed out.

Network interface Health

Backwards Compatibility Requirements

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description

bck-025		UT-0365	• Configure an MR client • Configure an MR OSS • Configure a non-MR MDS/MGS • create an FS • Run IO tests • Make sure that MR clients/OSS integrates seamlessly in the system.

Performance Requirements

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description
		UT-0370	Testing reconnects. In Large clusters it is possible that servers might need to handle a burst of client connects. The performance of such scenarios needs to be quantified.

Misc Error Scenarios

Primary Requirement ID	Secondary Requirement ID	Unit Test ID	Unit Test Description
			• mount an File System • delete all networks from the MDS
			• mount an File System • delete all networks from the MDS
			• mount an File System • delete all networks from the MDS
			• mount an FS • reboot the MDS • mount the MDS again
			• mount an FS • reboot the OSS • mount the OSS again
			• mount an FS • reboot the Client • mount the Client again
			• mount an FS with no modprobe.conf configured • the tcp network with a default tcp interface should be configured • add the same interface again via DLC • This operation should be a no-op