

IBM Spectrum LSF for SAS
Version 10 Release 1.0

Configuration Reference



Contents

Chapter 1. Configuration files.....	1
cshrc.lsf and profile.lsf.....	1
Set the LSF environment.....	2
Environment variables set by cshrc.lsf and profile.lsf.....	3
EGO environment variables set by cshrc.lsf and profile.lsf.....	5
hosts.....	9
install.config.....	12
Parameters.....	12
lim.acct.....	24
lsb.acct.....	24
Records and fields.....	25
lsb.applications.....	32
#INCLUDE.....	33
ABS_RUNLIMIT.....	33
BIND_JOB.....	33
CHKPNT_DIR.....	34
CHKPNT_INITPERIOD.....	34
CHKPNT_PERIOD.....	35
CHKPNT_METHOD.....	35
CHUNK_JOB_SIZE.....	35
CONTAINER.....	36
CORELIMIT.....	40
CPU_FREQUENCY.....	40
CPULIMIT.....	40
DATALIMIT.....	41
DESCRIPTION.....	41
DJOB_COMMFAIL_ACTION.....	42
DJOB_DISABLED.....	42
DJOB_ENV_SCRIPT.....	42
DJOB_HB_INTERVAL.....	43
DJOB_RESIZE_GRACE_PERIOD.....	43
DJOB_RU_INTERVAL.....	43
DOCKER_IMAGE_AFFINITY.....	44
ELIGIBLE_PEND_TIME_LIMIT.....	45
ENV_VARS.....	45
ESTIMATED_RUNTIME.....	46
EXEC_DRIVER.....	46
FILELIMIT.....	47
GPU_REQ.....	47
HOST_POST_EXEC.....	50
HOST_PRE_EXEC.....	50
JOB_CWD.....	51
JOB_CWD_TTL.....	51
JOB_INCLUDE_POSTPROC.....	52
JOB_POSTPROC_TIMEOUT.....	52
JOB_PREPROC_TIMEOUT.....	53
JOB_SIZE_LIST.....	53
JOB_STARTER.....	54
LOCAL_MAX_PREEEXEC_RETRY.....	54
LOCAL_MAX_PREEEXEC_RETRY_ACTION.....	55
MAX_JOB_PREEMPT.....	55

MAX_JOB_REQUEUE.....	56
MAX_PREEEXEC_RETRY.....	56
MAX_TOTAL_TIME_PREEMPT.....	56
MEMLIMIT.....	57
MEMLIMIT_TYPE.....	58
MIG.....	58
NAME.....	59
NETWORK_REQ.....	59
NICE.....	61
NO_PREEMPT_INTERVAL.....	62
NO_PREEMPT_FINISH_TIME.....	62
NO_PREEMPT_RUN_TIME.....	63
PEND_TIME_LIMIT.....	63
PERSISTENT_HOST_ORDER.....	64
PLAN.....	64
POST_EXEC.....	64
PREEMPT_DELAY.....	66
PRE_EXEC.....	66
PROCESSLIMIT.....	67
PRIORITY.....	67
RC_ACCOUNT.....	68
REMOTE_MAX_PREEEXEC_RETRY.....	68
REQUEUE_EXIT_VALUES.....	69
RERUNNABLE.....	70
RES_REQ.....	70
RESIZABLE_JOBS.....	74
RESIZE_NOTIFY_CMD.....	75
RESUME_CONTROL.....	75
RTASK_GONE_ACTION.....	76
RUNLIMIT.....	76
RUNTIME.....	77
STACKLIMIT.....	77
SUCCESS_EXIT_VALUES.....	77
SUSPEND_CONTROL.....	78
SWAPLIMIT.....	79
TASKLIMIT.....	79
TERMINATE_CONTROL.....	80
THREADLIMIT.....	81
USE_PAM_CREDS.....	81
WATCHDOG.....	81
Automatic time-based configuration.....	82
lsb.events.....	83
lsb.events structure.....	83
Limiting the size of lsb.events.....	83
Records and fields.....	83
lsb.globalpolicies.....	124
GlobalFairshare section.....	124
lsb.hosts.....	126
#INCLUDE.....	126
Host section.....	127
HostGroup section.....	131
HostPartition section.....	135
ComputeUnit section.....	137
Automatic time-based configuration.....	141
lsb.modules.....	141
After you change lsb.modules configuration.....	141
PluginModule section.....	141
lsb.params.....	144

Changing lsb.params configuration.....	144
Automatic time-based configuration.....	145
Parameters section.....	145
ABS_RUNLIMIT.....	145
ACCT_ARCHIVE_AGE.....	146
ACCT_ARCHIVE_SIZE.....	146
ACCT_ARCHIVE_TIME.....	147
ADVRSV_USER_LIMIT.....	147
ALLOCATION_PLANNER.....	147
ALLOW_EVENT_TYPE.....	148
AR_AVAILABLE_STATUS.....	149
AR_RUNLIMIT_CHECK.....	149
BJOBS_RES_REQ_DISPLAY.....	150
BSWITCH_MODIFY_RUSAGE.....	150
CHUNK_JOB_DURATION.....	151
CLEAN_PERIOD.....	151
COMMITTED_RUN_TIME_FACTOR.....	152
CLEAN_PERIOD_DONE.....	152
COMPUTE_UNIT_TYPES.....	152
CONDENSE_PENDING_REASONS.....	153
CPU_TIME_FACTOR.....	153
CSM_VALID_SMT.....	154
DEFAULT_APPLICATION.....	154
DEFAULT_BWAIT_TIMEOUT.....	155
DEFAULT_HOST_SPEC.....	155
DEFAULT_JOB_CWD.....	155
DEFAULT_JOB_OUTDIR.....	156
DEFAULT_JOBGROUP.....	157
DEFAULT_PROJECT.....	158
DEFAULT_QUEUE.....	158
DEFAULT_RESREQ_ORDER.....	159
DEFAULT_SLA_VELOCITY.....	159
DEFAULT_USER_GROUP.....	159
DETECT_IDLE_JOB_AFTER.....	160
DIAGNOSE_LOGDIR.....	160
DISABLE_UACCT_MAP.....	161
DOCKER_IMAGE_AFFINITY.....	161
EADMIN_TRIGGER_DURATION.....	162
EADMIN_TRIGGER_INTERVAL.....	162
EGO_SLOTBASED_VELOCITY_SLA.....	162
EGROUP_UPDATE_INTERVAL.....	163
ELIGIBLE_PENDINFO_SNAPSHOT_INTERVAL.....	163
ENABLE_DEFAULT_EGO_SLA.....	164
ENABLE_DIAGNOSE.....	164
ENABLE_EVENT_STREAM.....	164
ENABLE_EXIT_RATE_PER_SLOT.....	165
ENABLE_GPU_HIST_RUN_TIME.....	165
ENABLE_HIST_RUN_TIME.....	165
ENABLE_HOST_INTERSECTION.....	166
ENABLE_JOB_INFO_BY_ADMIN_ROLE.....	166
ENABLE_USER_RESUME.....	166
ENFORCE_ONE_UG_LIMITS.....	167
ENFORCE_UG_TREE.....	168
ESTIMATED_RUNTIME.....	168
ESTIMATOR_CPUS.....	169
ESTIMATOR_MAX_JOBS_PREDICTION.....	169
ESTIMATOR_MAX_RUNTIME_PREDICTION.....	170
ESTIMATOR_MAX_TIME_PREDICTION.....	170

ESTIMATOR_SIM_START_INTERVAL.....	171
EVALUATE_JOB_DEPENDENCY	172
EVALUATE_JOB_DEPENDENCY_TIMEOUT	173
EVALUATE_WAIT_CONDITION_TIMEOUT.....	174
EVENT_STREAM_FILE	174
EVENT_UPDATE_INTERVAL.....	174
EXIT_RATE_TYPE.....	175
EXTEND_JOB_EXCEPTION_NOTIFY.....	175
FAIRSHARE_ADJUSTMENT_FACTOR.....	176
FWD_JOB_FACTOR.....	176
GLOBAL_EXIT_RATE.....	177
GPU_REQ_MERGE.....	177
GPU_RUN_TIME_FACTOR.....	178
HIST_HOURS.....	178
JOB_ACCEPT_INTERVAL.....	179
JOB_ARRAY_EVENTS_COMBINE.....	179
JOB_ATTA_DIR.....	180
JOB_CWD_TTL.....	181
JOB_DEP_LAST_SUB.....	181
JOB_DISTRIBUTE_ON_HOST.....	182
JOB_EXIT_RATE_DURATION.....	182
JOB_GROUP_CLEAN.....	183
JOB_INCLUDE_POSTPROC.....	183
JOB_INFO_EVENT_DUMP_INTERVAL.....	184
JOB_INFO_MEMORY_CACHE_SIZE.....	184
JOB_POSITION_CONTROL_BY_ADMIN.....	185
JOB_POSTPROC_TIMEOUT.....	185
JOB_PREPROC_TIMEOUT.....	185
JOB_PRIORITY_OVER_TIME.....	186
JOB_RUNLIMIT_RATIO.....	187
JOB_SCHEDULING_INTERVAL.....	188
JOB_SPOOL_DIR.....	188
JOB_SWITCH2_EVENT (OBSOLETE).....	190
JOB_TERMINATE_INTERVAL.....	190
KILL_JOBS_OVER_RUNLIMIT.....	190
LOCAL_MAX_PREEXEC_RETRY.....	191
LOCAL_MAX_PREEXEC_RETRY_ACTION.....	191
LSB_SYNC_HOST_STAT_LIM.....	192
MAX_ACCT_ARCHIVE_FILE	192
MAX_CONCURRENT_QUERY.....	193
MAX_EVENT_STREAM_FILE_NUMBER.....	193
MAX_EVENT_STREAM_SIZE.....	194
MAX_INFO_DIRS.....	194
MAX_JOB_ARRAY_SIZE.....	195
MAX_JOB_ATTA_SIZE.....	195
MAX_JOB_NUM.....	195
MAX_JOB_PREEMPT.....	196
MAX_JOB_PREEMPT_RESET.....	196
MAX_JOB_QUEUE.....	196
MAX_JOBID.....	197
MAX_JOBINFO_QUERY_PERIOD.....	197
MAX_PEND_JOBS.....	198
MAX_PEND_SLOTS.....	198
MAX_PREEXEC_RETRY.....	199
MAX_PROTOCOL_INSTANCES.....	199
MAX_SBD_CONNS.....	199
MAX_SBD_FAIL.....	200
MAX_TOTAL_TIME_PREEMPT.....	200

MAX_USER_PRIORITY.....	201
MBD_EGO_CONNECT_TIMEOUT.....	201
MBD_EGO_READ_TIMEOUT.....	201
MBD_EGO_TIME2LIVE.....	202
MBD_QUERY_CPUS.....	202
MBD_REFRESH_TIME.....	203
MBD_SLEEP_TIME.....	204
MBD_USE_EGO_MXJ.....	204
MC_PENDING_REASON_PKG_SIZE.....	205
MC_PENDING_REASON_UPDATE_INTERVAL.....	205
MC_PLUGIN_SCHEDULE_ENHANCE.....	205
MC_PLUGIN_UPDATE_INTERVAL.....	206
MC_RECLAIM_DELAY.....	206
MC_RESOURCE_MATCHING_CRITERIA.....	207
MC_RUSAGE_UPDATE_INTERVAL.....	207
MC_SORT_BY_SUBMIT_TIME.....	208
MIN_SWITCH_PERIOD.....	208
NEWJOB_REFRESH.....	209
NO_PREEMPT_FINISH_TIME.....	209
NO_PREEMPT_INTERVAL.....	210
NO_PREEMPT_RUN_TIME.....	210
MAX_JOB_MSG_NUM.....	211
ORPHAN_JOB_TERM_GRACE_PERIOD.....	211
PARALLEL_SCHED_BY_SLOT.....	211
PEND_REASON_UPDATE_INTERVAL.....	212
PERFORMANCE_THRESHOLD_FILE.....	212
PERSIST_LIVE_CONFIG.....	212
PG_SUSP_IT.....	213
PLAN.....	213
POWER_ON_WAIT.....	214
POWER_RESET_CMD.....	214
POWER_RESUME_CMD.....	214
POWER_STATUS_LOG_MAX.....	215
POWER_SUSPEND_CMD.....	215
POWER_SUSPEND_TIMEOUT.....	215
PREEMPT_DELAY.....	216
PREEMPT_FOR.....	216
PREEMPT_JOBTYPE.....	218
PREEMPTABLE_RESOURCES.....	218
PREEMPTION_WAIT_TIME.....	219
PREEXEC_EXCLUDE_HOST_EXIT_VALUES.....	219
PRIVILEGED_USER_FORCE_BKILL.....	220
RC_MAX_REQUESTS.....	220
RELAX_JOB_DISPATCH_ORDER.....	220
REMOVE_HUNG_JOBS_FOR.....	221
REMOTE_MAX_PREEXEC_RETRY.....	222
RESIZABLE_JOBS.....	222
RESOURCE_RESERVE_PER_TASK.....	223
RUN_JOB_FACTOR.....	223
RUN_TIME_DECAY.....	224
RUN_TIME_FACTOR.....	224
SBD_SLEEP_TIME.....	224
SCHED_METRIC_ENABLE.....	225
SCHED_METRIC_SAMPLE_PERIOD.....	225
SCHED_PER_JOB_SORT.....	225
SCHEDULER_THREADS.....	226
SECURE_INFODIR_USER_ACCESS.....	227
SECURE_JOB_INFO_LEVEL.....	228

SLA_TIMER.....	229
SSCHED_ACCT_DIR.....	230
SSCHED_MAX_RUNLIMIT.....	230
SSCHED_MAX_TASKS.....	230
SSCHED_REQUEUE_LIMIT.....	231
SSCHED_RETRY_LIMIT.....	231
SSCHED_UPDATE_SUMMARY_BY_TASK.....	231
SSCHED_UPDATE_SUMMARY_INTERVAL.....	232
STRICT_UG_CONTROL.....	232
STRIPING_WITH_MINIMUM_NETWORK.....	233
SUB_TRY_INTERVAL.....	233
SYSTEM_MAPPING_ACCOUNT.....	234
TRACK_ELIGIBLE_PENDINFO.....	234
USE_SUSP_SLOTS.....	235
lsb.queues.....	235
#INCLUDE.....	236
ADMINISTRATORS.....	236
APS_PRIORITY.....	236
BACKFILL.....	239
CHKPNT.....	240
CHUNK_JOB_SIZE.....	241
COMMITTED_RUN_TIME_FACTOR.....	241
CONTAINER.....	242
CORELIMIT.....	245
CPU_FREQUENCY.....	246
CPULIMIT.....	246
CPU_TIME_FACTOR.....	247
CSM_REQ.....	247
DATALIMIT.....	247
DATA_TRANSFER.....	248
DEFAULT_EXTSCHED.....	249
DEFAULT_HOST_SPEC.....	249
DESCRIPTION.....	249
DISPATCH_BY_QUEUE.....	249
DISPATCH_ORDER.....	250
DISPATCH_WINDOW.....	250
DOCKER_IMAGE_AFFINITY.....	251
ELIGIBLE_PEND_TIME_LIMIT.....	251
ENABLE_GPU_HIST_RUN_TIME.....	252
ENABLE_HIST_RUN_TIME.....	252
ESTIMATED_RUNTIME.....	252
EXCLUSIVE.....	253
EXEC_DRIVER.....	253
EXTENDABLE_RUNLIMIT.....	254
FAIRSHARE.....	255
FAIRSHARE_ADJUSTMENT_FACTOR.....	256
FAIRSHARE_QUEUES.....	256
FILELIMIT.....	257
FWD_JOB_FACTOR.....	257
FWD_USERS.....	258
GPU_REQ.....	258
GPU_RUN_TIME_FACTOR.....	260
HIST_HOURS.....	261
HJOB_LIMIT.....	261
HOST_POST_EXEC.....	262
HOST_PRE_EXEC.....	262
HOSTLIMIT_PER_JOB.....	263
HOSTS.....	263

IGNORE_DEADLINE.....	266
IMPT_JOBKLG.....	266
IMPT_TASKKLG.....	266
INTERACTIVE.....	267
INTERRUPTIBLE_BACKFILL.....	267
JOB_ACCEPT_INTERVAL.....	268
JOB_ACTION_WARNING_TIME.....	268
JOB_CONTROLS.....	269
JOB_IDLE.....	270
JOB_OVERRUN.....	270
JOB_SIZE_LIST.....	271
JOB_STARTER.....	271
JOB_UNDERRUN.....	272
JOB_WARNING_ACTION.....	272
LOAD_INDEX.....	273
LOCAL_MAX_PREEEXEC_RETRY.....	274
LOCAL_MAX_PREEEXEC_RETRY_ACTION.....	274
MANDATORY_EXTSCHED.....	275
MAX_JOB_PREEMPT.....	275
MAX_JOB_REQUEUE.....	275
MAX_PREEEXEC_RETRY.....	275
MAX_PROTOCOL_INSTANCES.....	276
MAX_RSCHED_TIME.....	276
MAX_SLOTS_IN_POOL.....	277
MAX_TOTAL_TIME_PREEMPT.....	277
MEMLIMIT.....	278
MIG.....	279
NETWORK_REQ.....	279
NEW_JOB_SCHED_DELAY.....	282
NICE.....	282
NO_PREEMPT_INTERVAL.....	282
PEND_TIME_LIMIT.....	283
PJOB_LIMIT.....	283
PLAN.....	284
POST_EXEC.....	284
PRE_EXEC.....	285
PREEMPTION.....	286
PREEMPT_DELAY.....	288
PRIORITY.....	288
PROCESSLIMIT.....	289
QJOB_LIMIT.....	289
QUEUE_GROUP.....	289
QUEUE_NAME.....	290
RC_ACCOUNT.....	290
RC_DEMAND_POLICY.....	290
RC_HOSTS.....	291
RCVJOBS_FROM.....	292
RELAX_JOB_DISPATCH_ORDER.....	293
REMOTE_MAX_PREEEXEC_RETRY.....	293
REQUEUE_EXIT_VALUES.....	294
RERUNNABLE.....	295
RESOURCE_RESERVE.....	295
RES_REQ.....	296
RESRSV_LIMIT.....	298
RESUME_COND.....	299
RUN_JOB_FACTOR.....	299
RUN_TIME_DECAY.....	300
RUN_TIME_FACTOR.....	300

RUN_WINDOW.....	300
RUNLIMIT.....	301
SLA_GUARANTEES_IGNORE.....	302
SLOT_POOL.....	302
SLOT_RESERVE.....	302
SLOT_SHARE.....	303
SNDJOBS_TO.....	304
STACKLIMIT.....	305
STOP_COND.....	305
SUCCESS_EXIT_VALUES.....	305
SWAPLIMIT.....	306
TASKLIMIT.....	306
TERMINATE_WHEN.....	307
THREADLIMIT.....	308
UJOB_LIMIT.....	308
USE_PAM_CREDS.....	308
USE_PRIORITY_IN_POOL.....	309
USERS.....	309
Automatic time-based configuration.....	310
lsb.reasons.....	310
#INCLUDE.....	310
CustomPendingReason section.....	311
lsb.resources.....	312
#INCLUDE.....	312
Limit section.....	312
GuaranteedResourcePool section.....	330
HostExport section.....	335
SharedResourceExport section.....	338
ResourceReservation section.....	339
ReservationUsage section.....	342
Automatic time-based configuration.....	344
PowerPolicy section.....	344
lsb.serviceclasses.....	347
ACCESS_CONTROL.....	348
AUTO_ATTACH.....	349
CONSUMER.....	350
CONTROL_ACTION.....	350
DESCRIPTION.....	351
EGO_RES_REQ.....	351
EGO_RESOURCE_GROUP.....	351
GOALS.....	352
MAX_HOST_IDLE_TIME.....	354
NAME.....	354
PRIORITY.....	354
USER_GROUP.....	355
Examples.....	355
lsb.threshold.....	357
lsb.users.....	358
#INCLUDE.....	358
UserGroup section.....	358
User section.....	362
UserMap section.....	364
Automatic time-based configuration.....	365
lsf.acct.....	365
lsf.cluster.....	367
Parameters section.....	368
ClusterAdmins section.....	377
Host section.....	378

ResourceMap section.....	382
RemoteClusters section.....	383
lsf.conf.....	385
DAEMON_SHUTDOWN_DELAY.....	386
EBROKERD_HOST_CLEAN_DELAY.....	386
EGO_DAEMONS_CPUS.....	387
EGO_DEFINE_NCPUS.....	388
EGO_ELIM_DIR.....	388
EGO_ENABLE_AUTO_DAEMON_SHUTDOWN.....	389
EGO_ESLIM_TIMEOUT.....	389
EGO_LIM_PG_SMOOTH_FACTOR.....	389
EGO_PIM_SWAP_REPORT.....	390
JOB_STARTER_EXTEND.....	390
LS_DUPLICATE_RESOURCE_CHECKING.....	391
LSB_AFS_BIN_DIR.....	391
LSB_AFS_JOB_SUPPORT.....	391
LSB_AFS_LIB_DIR.....	392
LSB_API_CONNTIMEOUT.....	392
LSB_API_RECVTIMEOUT.....	393
LSB_API_VERBOSE.....	393
LSB_BHOSTS_FORMAT.....	393
LSB_BJOBS_CONSISTENT_EXIT_CODE.....	395
LSB_BJOBS_FORMAT.....	397
LSB_BJOBS_PENDREASON_LEVEL.....	404
LSB_BLOCK_JOBINFO_TIMEOUT.....	404
LSB_BMGROUP_ALLREMOTE_EXPAND.....	405
LSB_BPEEK_REMOTE_OUTPUT.....	405
LSB_BSUB_ERR_RETRY.....	405
LSB_BSUB_PARSE_SCRIPT.....	406
LSB_BUSERS_FORMAT.....	407
LSB_BWAIT_REREG_INTERVAL.....	408
LSB_CALL_LIM_RETRY.....	408
LSB_CGROUP_MEM_INCLUDE_CACHE.....	409
LSB_CHECK_JOB_PID_REUSE_ON_REBOOT.....	409
LSB_CHUNK_RUSAGE.....	409
LSB_CMD_LOG_MASK.....	410
LSB_CMD_LOGDIR.....	411
LSB_CONFDIR.....	411
LSB_CPUSET_BESTCPUS.....	411
LSB_CPUSET_DISPLAY_CPULIST.....	412
LSB_DEBUG.....	412
LSB_DEBUG_CMD.....	413
LSB_DEBUG_GPD.....	414
LSB_DEBUG_MBD.....	415
LSB_DEBUG_SBD.....	416
LSB_DEBUG_SCH.....	416
LSB_DISABLE_LIMLOCK_EXCL.....	417
LSB_DISABLE_RERUN_POST_EXEC.....	417
LSB_DISABLE_SUSPEND_AR_JOBS.....	418
LSB_DISPATCH_CHECK_RESUME_INTVL.....	418
LSB_DISPLAY_YEAR.....	418
LSB_EAUTH_DATA_REUSE.....	419
LSB_EAUTH_EACH_SUBPACK.....	419
LSB_ECHKPNT_KEEP_OUTPUT.....	419
LSB_ECHKPNT_METHOD.....	420
LSB_ECHKPNT_METHOD_DIR.....	420
LSB_ENABLE_ESTIMATION.....	421
LSB_ENABLE_HPC_ALLOCATION.....	421

LSB_ENABLE_PERF_METRICS_LOG.....	422
LSB_ESUB_METHOD.....	422
LSB_ESUB_SINGLE_QUOTE.....	423
LSB_ESWITCH_METHOD.....	423
LSB_EVENTS_FILE_KEEP_OPEN.....	424
LSB_EXCLUDE_HOST_PERIOD.....	424
LSB_FANOUT_TIMEOUT_PER_LAYER.....	425
LSB_FORK_JOB_REQUEST.....	425
LSB_GPD_CLUSTER.....	425
LSB_GPD_PORT.....	425
LSB_GPU_AUTOBOOST (Obsolete).....	426
LSB_GPU_NEW_SYNTAX.....	426
LSB_GPU_REQ.....	427
LSB_GPU_POWEROFF_DURATION.....	429
LSB_GSLA_DISPLAY_ALLOC_HOSTS.....	430
LSB_GSLA_PREFER_ADRSV_HOST.....	430
LSB_HJOB_PER_SESSION.....	431
LSB_HMS_TIME_FORMAT.....	431
LSB_HUGETLB.....	432
LSB_IGNORE_ELIM_RES.....	433
LSB_INDEX_BY_JOB.....	433
LSB_INTERACT_MSG_ENH.....	433
LSB_INTERACT_MSG_INTVAL.....	434
LSB_JOB_CPULIMIT.....	434
LSB_JOBID_DISP_LENGTH.....	435
LSB_JOB_MEMLIMIT.....	436
LSB_JOB_OUTPUT_LOGGING.....	437
LSB_JOB_REPORT_MAIL.....	438
LSB_JOB_SCRIPT_TRAP_SIGNALS.....	438
LSB_JOB_TMPDIR.....	438
LSB_JOBID_DISP_LENGTH.....	440
LSB_JOBINFO_DIR.....	440
LSB_JSM_DEFAULT.....	441
LSB_KEEP_SYSDEF_RLIMIT.....	441
LSB_KRB_AIX_LOAD_IBM_NAS.....	442
LSB_KRB_CHECK_INTERVAL.....	442
LSB_KRB_LIB_PATH.....	442
LSB_KRB_RENEW_MARGIN.....	443
LSB_KRB_TGT_FWD.....	443
LSB_KRB_TGT_DIR.....	443
LSB_LOCALDIR.....	444
LSB_LOG_MASK_MBD.....	444
LSB_LOG_MASK_SBD.....	445
LSB_LOG_MASK_SCH.....	445
LSB_MAIL_FROM_DOMAIN.....	446
LSB_MAIL_TIMESTAMP_EPOCH_SECONDS.....	446
LSB_MAILPROG.....	446
LSB_MAILSENDER.....	447
LSB_MAILSERVER.....	448
LSB_MAILSIZE_LIMIT.....	448
LSB_MAILTO.....	449
LSB_MAX_ASKED_HOSTS_NUMBER.....	450
LSB_MAX_FORWARD_PER_SESSION.....	450
LSB_MAX_JOB_DISPATCH_PER_SESSION.....	450
LSB_MAX_PACK_JOBS.....	451
LSB_MAX_PROBE_SBD.....	451
LSB_MBD_BUSY_MSG.....	452
LSB_MBD_CONNECT_FAIL_MSG.....	452

LSB_MBD_DOWN_MSG.....	453
LSB_MBD_PORT	453
LSB_MBD_QUERY_RETRY_MSG.....	453
LSB_MC_CHKPNT_RERUN.....	453
LSB_MC_DISABLE_HOST_LOOKUP.....	454
LSB_MC_INITFAIL_MAIL.....	454
LSB_MC_INITFAIL_RETRY.....	454
LSB_MC_RECALL_EXIT_INFO.....	455
LSB_MC_RECALLED_JOBS_HIDE.....	455
LSB_MEMLIMIT_ENFORCE.....	456
LSB_MEMLIMIT_ENF_CONTROL	456
LSB_MIG2PEND.....	457
LSB_MIXED_PATH_DELIMITER.....	457
LSB_MIXED_PATH_ENABLE.....	458
LSB_MOD_ALL_JOBS.....	459
LSB_NCPU_ENFORCE.....	459
LSB_NQS_PORT.....	459
LSB_NUM_NIOS_CALLBACK_THREADS.....	460
LSB_PACK_MESUB.....	460
LSB_PACK_SKIP_ERROR.....	461
LSB_PERF_METRICS_LOGDIR.....	461
LSB_PERF_METRICS_SAMPLE_PERIOD.....	461
LSB_PLAN_KEEP_RESERVE.....	461
LSB_POSTEXEC_SEND_MAIL.....	462
LSB_PROFILE_MBD	462
LSB_PROFILE_SCH	463
LSB_PROJECT_LIMITS.....	463
LSB_QUERY_ENH.....	464
LSB_QUERY_PORT.....	464
LSB_QUERY_RETRY_INTERVAL_STEP.....	465
LSB_RC_DEFAULT_HOST_TYPE.....	465
LSB_RC_EXTERNAL_HOST_FLAG.....	466
LSB_RC_EXTERNAL_HOST_IDLE_TIME.....	466
LSB_RC_EXTERNAL_HOST_MAX_TTL.....	467
LSB_RC_MQTT_ERROR_LIMIT.....	467
LSB_RC_QUERY_INTERVAL.....	467
LSB_RC_TEMPLATE_REQUEST_DELAY.....	468
LSB_RC_UPDATE_INTERVAL.....	468
LSB_REQUEUE_TO_BOTTOM.....	469
LSB_RESOURCE_ENFORCE.....	469
LSB_RLA_PORT.....	471
LSB_RLA_UPDATE.....	471
LSB_RLA_WORKDIR.....	471
LSB_SACCT_ONE_UG.....	471
LSB_SBD_PORT.....	472
LSB_SBD_STARTUP_RETRY_INTERVAL.....	472
LSB_SBD_STARTUP_RETRY_TIMES.....	472
LSB_SET_TMPDIR.....	473
LSB_SHAREDIR.....	473
LSB_SHORT_HOSTLIST.....	474
LSB_SIGSTOP.....	474
LSB_SKIP_FULL_HOSTS.....	474
LSB_SSH_XFORWARD_CMD.....	475
LSF_STAGE_IN_EXEC.....	475
LSB_STAGE_MAX_STAGE_IN.....	476
LSB_STAGE_OUT_EXEC.....	476
LSB_STAGE_STORAGE.....	476
LSB_STAGE_TRANSFER_RATE.....	477

LSB_STEP_CGROUP_DEFAULT.....	477
LSB_START_EBROKERD.....	477
LSB_START_MPS.....	478
LSB_STDOUT_DIRECT.....	478
LSB_STOP_IGNORE_IT.....	479
LSB_SUB_COMMANDNAME.....	479
LSB_SUBK_SHOW_EXEC_HOST.....	480
LSB_SUPPRESS_CUSTOM_REASONS.....	480
LSB_TERMINAL_SERVICE_PORT.....	480
LSB_TIME_CMD.....	481
LSB_TIME_DMD.....	481
LSB_TIME_GPD.....	481
LSB_TIME_MBD.....	482
LSB_TIME_SCH.....	482
LSB_TIME_RESERVE_NUMJOBS.....	482
LSB_TIME_SBD.....	483
LSB_TSJOBS_HELPER_HOSTS.....	483
LSB_TSJOBS_HELPER_PORT.....	484
LSB_TSJOBS_HELPER_TIMEOUT.....	484
LSB_USE_SEQUENTIAL_SIGNAL.....	484
LSB_USER_REQUEUE_TO_BOTTOM.....	485
LSB_UTMP.....	485
LSF_AC_PNC_URL.....	486
LSF_AM_OPTIONS.....	486
LSF_API_CONNTIMEOUT.....	486
LSF_API_RECVMTIMEOUT.....	487
LSF_ASPLUGIN.....	487
LSF_AUTH.....	487
LSF_AUTH_DAEMONS.....	488
LSF_BIND_JOB.....	488
LSF_BINDIR.....	489
LSF_BMPLUGIN.....	489
LSF_CLEARCASE_WRAPPER_RETRY.....	489
LSF_CLEARCASE_WRAPPER_RETRY_INTERVAL.....	490
LSF_CMD_LOG_MASK.....	491
LSF_CMD_LOGDIR.....	491
LSF_COLLECT_ENERGY_USAGE.....	492
LSF_CONF_RETRY_INT.....	492
LSF_CONF_RETRY_MAX.....	492
LSF_CONFDIR.....	493
LSF_CONNECTION_CHANGE.....	493
LSF_CPUSETLIB.....	494
LSF_CRASH_LOG.....	494
LSF_CRAY_RELEASE_TIMEOUT.....	494
LSF_CRAY_RUR_DIR.....	495
LSF_CRAY_RUR_EPILOG_PATH.....	495
LSF_CRAY_RUR_PROLOG_PATH.....	495
LSF_DATA_BSUB_CHKSUM.....	496
LSF_DATA_CLUSTER.....	496
LSF_DATA_HOSTS.....	497
LSF_DATA_NO_SSH_CHK_HOSTS.....	497
LSF_DATA_PORT.....	498
LSF_DATA_SCP_CMD.....	498
LSF_DATA_SKIP_GROUP_CHECK.....	499
LSF_DATA_SSH_CMD.....	499
LSF_DAEMON_WRAP.....	500
LSF_DAEMONS_CPUS.....	500
LSF_DCGM_PORT.....	501

LSF_DEBUG_CMD	502
LSF_DEBUG_LIM.....	503
LSF_DEBUG_RES.....	505
LSF_DEFAULT_FREQUENCY.....	505
LSF_DHCP_ENV.....	506
LSF_DISABLE_LSRUN.....	506
LSF_DISCARD_LOG.....	507
LSF_DISPATCHER_LOGDIR.....	507
LSF_DJOB_TASK_REG_WAIT_TIME.....	508
LSF_DNS_CACHE.....	508
LSF_DUALSTACK_PREFER_IPV6.....	509
LSF_DYNAMIC_HOST_TIMEOUT.....	509
LSF_DYNAMIC_HOST_WAIT_TIME.....	510
LSF_EGO_DAEMON_CONTROL.....	511
LSF_EGO_ENVDIR.....	511
LSF_ELIM_DIR.....	512
LSF_ENABLE_BEAT_SERVICE.....	512
LSF_ENABLE_EGO.....	513
LSF_ENABLE_EXTSCHEDULER.....	514
LSF_ENABLE_SUPPORT_IPV6.....	514
LSF_ENABLE_TMP_UNIT.....	514
LSF_ENV_OVERRIDE.....	515
LSF_ENVDIR.....	515
LSF_EVENT_PROGRAM.....	516
LSF_EVENT_RECEIVER.....	516
LSF_EXT_SERVERDIR.....	516
LSF_GET_CONF.....	517
LSF_GPU_AUTOCONFIG.....	517
LSF_HOST_CACHE_DISABLE.....	517
LSF_HOST_CACHE_NTTL.....	518
LSF_HOST_CACHE_PTTL.....	518
LSF_HOST_MAX_GPU.....	519
LSF_HPC_EXTENSIONS.....	519
LSF_HPC_PJL_LOADENV_TIMEOUT.....	522
LSF_HWLOC_DYNAMIC.....	523
LSF_ID_PORT	523
LSF_IMAGE_INFO_EXPIRE_INTERVAL.....	523
LSF_IMAGE_INFO_PUBLISH_INTERVAL.....	524
LSF_INCLUDEDIR.....	524
LSF_INDEP.....	524
LSF_INTELLIGENT_CPU_BIND.....	525
LSF_INTERACTIVE_STDERR.....	526
LSF_LD_SECURITY.....	527
LSF_LIBDIR.....	528
LSF_LIC_SCHED_CLUSTER_ALIAS.....	528
LSF_LIC_SCHED_HOSTS.....	528
LSF_LIC_SCHED_PREEMPT_QUEUE.....	529
LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE.....	529
LSF_LIC_SCHED_PREEMPT_STOP.....	530
LSF_LIC_SCHED_STRICT_PROJECT_NAME.....	530
LSF_LIM_API_NTRIES.....	530
LSF_LIM_DEBUG.....	531
LSF_LIM_IGNORE_CHECKSUM.....	531
LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT	532
LSF_LIM_STARTUP_RETRY_INTERVAL.....	532
LSF_LIM_STARTUP_RETRY_TIMES.....	533
LSF_LINUX_CGROUP_ACCT.....	533
LSF_LIVE_CONFDIR.....	533

LSF_LOAD_USER_PROFILE.....	534
LSF_LOCAL_RESOURCES.....	534
LSF_LOG_MASK.....	535
LSF_LOG_MASK_LIM.....	536
LSF_LOG_MASK_RES.....	537
LSF_LOG_MASK_WIN.....	538
LSF_LOG_QUEUE_SIZE.....	538
LSF_LOG_THREAD.....	539
LSF_LOGDIR.....	539
LSF_LOGDIR_USE_WIN_REG.....	540
LSF_LOGFILE_OWNER.....	541
LSF_LSHOSTS_FORMAT.....	541
LSF_LSLOAD_FORMAT.....	543
LSF_LSLOGIN_SSH.....	546
LSF_MACHDEP.....	546
LSF_MANAGE_FREQUENCY.....	546
LSF_MANDIR.....	547
LSF_MASTER_LIST.....	547
LSF_MASTER_NSLOOKUP_TIMEOUT.....	548
LSF_MAX_TRY_ADD_HOST.....	549
LSF_MC_FORWARD_FAIRSHARE_CHARGE_DURATION.....	549
LSF_MC_NON_PRIVILEGED_PORTS.....	549
LSF_MISC.....	550
LSF_MQ_BROKER_HOSTS.....	550
LSF_NIOS_DEBUG.....	551
LSF_NIOS_ERR_LOGDIR.....	551
LSF_NIOS_JOBSTATUS_INTERVAL.....	552
LSF_NIOS_MAX_TASKS.....	552
LSF_NIOS_PORT_RANGE.....	553
LSF_NIOS_RES_HEARTBEAT.....	553
LSF_NON_PRIVILEGED_PORTS.....	554
LSF_PAM_APPL_CHKPNP.....	554
LSF_PAM_CLEAN_JOB_DELAY.....	554
LSF_PAM_HOSTLIST_USE.....	555
LSF_PAM_PLUGINDIR.....	555
LSF_PAM_USE_ASH.....	556
LSF_PASSWD_DIR.....	556
LSF_PE_NETWORK_NUM.....	556
LSF_PE_NETWORK_UPDATE_INTERVAL.....	557
LSF_PIM_INFODIR.....	557
LSF_PIM_LINUX_ENHANCE.....	557
LSF_PIM_SLEEPTIME.....	558
LSF_PIM_SLEEPTIME_UPDATE.....	558
LSF_PLATFORM_COMPATIBILITY.....	559
LSF_POE_TIMEOUT_BIND.....	559
LSF_POE_TIMEOUT_SELECT.....	559
LSF_PROCESS_TRACKING.....	560
LSF_QUERY_ES_FUNCTIONS.....	560
LSF_QUERY_ES_SERVERS.....	561
LSF_RC_AUDIT_LOG.....	561
LSF_REG_FLOAT_HOSTS.....	562
LSF_REG_HOST_INTERVAL.....	563
LSF_REMOTE_COPY_CMD.....	563
LSF_REPLACE_PIM_WITH_LINUX_CGROUP.....	564
LSF_RES_ACCT.....	564
LSF_RES_ACCTDIR.....	565
LSF_RES_ACTIVE_TIME.....	565
LSF_RES_ALIVE_TIMEOUT.....	566

LSF_RES_CLIENT_TIMEOUT.....	566
LSF_RES_CONNECT_RETRY.....	566
LSF_RES_DEBUG.....	566
LSF_RES_PORT.....	567
LSF_RES_RLIMIT_UNLIM.....	567
LSF_RES_STARTUP_RETRY_INTERVAL.....	567
LSF_RES_STARTUP_RETRY_TIMES.....	568
LSF_RES_SYNCUP_INTERVAL.....	568
LSF_RES_TIMEOUT.....	569
LSF_ROOT_REX.....	569
LSF_RSH.....	569
LSF_SECUREDIR.....	570
LSF_SEND_CONFINFO_TCP_THRESHOLD.....	570
LSF_SERVER_HOSTS.....	571
LSF_SERVERDIR.....	572
LSF_SHELL_AT_USERS.....	572
LSF_SHIFT_JIS_INPUT.....	572
LSF_STRICT_CHECKING.....	573
LSF_STRICT_RESREQ.....	573
LSF_STRIP_DOMAIN.....	574
LSF_TIME_CMD.....	574
LSF_TIME_LIM.....	574
LSF_TIME_PIM.....	575
LSF_TIME_RES.....	575
LSF_TMPDIR.....	576
LSF_UGROUP_TRANSFER.....	576
LSF_UNIT_FOR_LIMITS.....	576
LSF_UDP_PORT_RANGE.....	578
LSF_USE_HOSTEQUIV.....	578
LSF_USER_DOMAIN.....	579
LSF_VPLUGIN.....	579
LSF_WINDOWS_HOST_TYPES.....	580
MQTT_BROKER_HOST.....	580
MQTT_BROKER_PORT.....	581
RC_MAX_AUDIT_LOG_KEEP_TIME.....	582
RC_MAX_AUDIT_LOG_SIZE.....	582
XLSF_APPDIR.....	582
XLSF_UIDDIR.....	582
MC_PLUGIN_REMOTE_RESOURCE.....	583
lsf.datamanager.....	583
lsf.datamanager Parameters section.....	584
RemoteDataManagers section.....	591
lsf.licensescheduler.....	592
Parameters section.....	592
Clusters section.....	611
ServiceDomain section.....	611
Feature section.....	615
FeatureGroup section.....	638
ProjectGroup section.....	639
Projects section.....	641
Automatic time-based configuration.....	643
lsf.shared.....	643
#INCLUDE.....	643
Cluster section.....	644
HostType section.....	644
HostModel section.....	645
Resource section.....	646
lsf.sudoers.....	648

LSB_PRE_POST_EXEC_USER.....	650
LSF_EAUTH_KEY	650
LSF_EAUTH_USER.....	651
LSF_EEXEC_USER.....	651
LSF_EGO_ADMIN_PASSWD.....	651
LSF_EGO_ADMIN_USER.....	652
LSF_LOAD_PLUGINS.....	652
LSF_STARTUP_PATH.....	652
LSF_STARTUP_USERS.....	653
lsf.task.....	653
Task files.....	654
Format of task files.....	655
setup.config.....	656
LS_ADMIN.....	657
LS_HOSTS.....	657
LS_LMSTAT_PATH.....	658
LS_TOP.....	658
slave.config.....	658
EGO_DAEMON_CONTROL.....	659
ENABLE_EGO.....	660
EP_BACKUP.....	660
LSF_ADMINS.....	661
LSF_ENTITLEMENT_FILE.....	661
LSF_LIM_PORT.....	662
LSF_SERVER_HOSTS.....	662
LSF_TARDIR.....	663
LSF_LOCAL_RESOURCES.....	663
LSF_TOP.....	664
SILENT_INSTALL.....	664
LSF_SILENT_INSTALL_TARLIST.....	664

Chapter 2. Environment variables..... 667

Environment variables set for job execution.....	667
Environment variables for resize notification command.....	668
Environment variables for session scheduler (ssched).....	668
Environment variables for data provenance.....	670
Environment variable reference.....	670
BSUB_BLOCK.....	670
BSUB_CHK_RESREQ.....	671
BSUB_QUIET.....	671
BSUB_QUIET2.....	672
BSUB_STDERR.....	672
CLEARCASE_DRIVE.....	672
CLEARCASE_MOUNTDIR.....	673
CLEARCASE_ROOT	674
DAEMON_WRAP_DEBUG.....	674
ELIM_ABORT_VALUE	675
LS_EXEC_T	675
LS_JOBPID.....	675
LS_SUBCWD	675
LSB_AFFINITY_HOSTFILE.....	676
LSB_BHIST_HOURS.....	677
LSB_BIND_CPU_LIST.....	677
LSB_BIND_MEM_LIST.....	677
LSB_BIND_MEM_POLICY.....	678
LSB_BHOSTS_FORMAT.....	678
LSB_BJOBS_FORMAT.....	678

LSB_BMGROUP_ALLREMOTE_EXPAND.....	678
LSB_BQUEUES_FORMAT.....	678
LSB_BSUB_ERR_RETRY.....	678
LSB_BUSERS_FORMAT.....	679
LSB_CHKPNT_DIR.....	679
LSB_CONTAINER_IMAGE_UPDATE.....	679
LSB_DATA_CACHE_TOP.....	680
LSB_DATA_META_FILE.....	680
LSB_DATA_PROVENANCE.....	680
LSB_DEBUG.....	680
LSB_DEBUG_CMD.....	681
LSB_DEBUG_MBD.....	681
LSB_DEBUG_SBD.....	681
LSB_DEBUG_SCH.....	681
LSB_DEFAULT_JOBGROUP.....	681
LSB_DEFAULTPROJECT.....	682
LSB_DEFAULTQUEUE.....	683
LSB_DJOB_COMMFAIL_ACTION.....	683
LSB_DJOB_ENV_SCRIPT.....	684
LSB_DJOB_HB_INTERVAL.....	684
LSB_DJOB_RU_INTERVAL.....	684
LSB_DJOB_PE_NETWORK.....	685
LSB_DJOB_NUMPROC.....	685
LSB_DJOB_RANKFILE.....	685
LSB_DJOB_TASK_BIND.....	686
LSB_DOCKER_IMAGE_AFFINITY.....	686
LSB_ECHKPNT_METHOD.....	687
LSB_ECHKPNT_METHOD_DIR.....	687
LSB_ECHKPNT_KEEP_OUTPUT.....	687
LSB_EFFECTIVE_RSRCREQ.....	687
LSB_ERESTART_USRCMD.....	687
LSB_EXEC_RUSAGE.....	688
LSB_EXECHOSTS.....	688
LSB_EXIT_IF_CWD_NOTEXIST.....	688
LSB_EXIT_PRE_ABORT.....	689
LSB_EXIT_REQQUEUE.....	689
LSB_FRAMES.....	690
LSB_HMS_TIME_FORMAT.....	690
LSB_HOSTS.....	690
LSB_INTERACTIVE.....	691
LSB_JOBAFF_REQ.....	691
LSB_JOB_INCLUDE_POSTPROC.....	691
LSB_JOBEXIT_INFO.....	692
LSB_JOBEXIT_STAT.....	692
LSB_JOBFILENAME.....	693
LSB_JOBGROUP.....	693
LSB_JOBID.....	693
LSB_JOBINDEX.....	693
LSB_JOBINDEX_STEP.....	694
LSB_JOBNAME.....	695
LSB_JOBPEND.....	695
LSB_JOBPGIDS.....	696
LSB_JOBPIDS.....	696
LSB_MAILSIZE.....	696
LSB_MAX_NUM_PROCESSORS.....	697
LSB_MCPU_HOSTS.....	697
LSB_NQS_PORT.....	698
LSB_NTRIES.....	698

LSB_OLD_JOBID.....	698
LSB_OUTDIR.....	699
LSB_OUTPUT_TARGETFAILED.....	699
LSB_PROJECT_NAME.....	699
LSB_QUEUE	700
LSB_RANK_HOSTFILE.....	700
LSB_REMOTEINDEX.....	700
LSB_REMOTEJID.....	701
LSB_JOB_REPORT_MAIL.....	701
LSB_RESTART	701
LSB_RESTART_PGID.....	702
LSB_RESTART_PID.....	702
LSB_RTASK_GONE_ACTION.....	703
LSB_SUB_ADDITIONAL.....	703
LSB_SUB_CLUSTER.....	703
LSB_SUB_COMMANDNAME.....	704
LSB_SUB_COMMAND_LINE.....	704
LSB_SUB_EXTSCHED_PARAM.....	704
LSB_SUB_JOB_ACTION_WARNING_TIME.....	704
LSB_SUB_JOB_ERR.....	704
LSB_SUB_JOB_ID.....	705
LSB_SUB_JOB_QUEUE	705
LSB_SUB_JOB_WARNING_ACTION.....	706
LSB_SUB_MEM_SWAP_HOST_LIMIT.....	706
LSB_SUB_PARM_FILE.....	706
LSB_SUB_RES_REQ.....	706
LSB_SUB_USER.....	707
LSB_SUCCESS_EXIT_VALUES.....	707
LSB_SUSP_REASONS.....	707
LSB_SUSP_SUBREASONS.....	707
LSB_SWITCH_ABORT_VALUE.....	708
LSB_SWITCH_MODIFY_FILE.....	709
LSB_SWITCH_PARM_FILE.....	709
LSB_TASK_GEOMETRY.....	709
LSB_UNIXGROUP.....	710
LSB_USER_BIND_CPU_LIST.....	710
LSB_USER_BIND_JOB.....	710
LSF_AC_JOB_NOTIFICATION.....	710
LSF_CMD_LOGDIR.....	710
LSF_DEBUG_CMD.....	711
LSF_DEBUG_LIM.....	711
LSF_DEBUG_RES.....	711
LSF_EAUTH_AUX_DATA.....	711
LSF_EAUTH_AUX_PASS.....	711
LSF_EAUTH_CLIENT.....	711
LSF_EAUTH_SERVER.....	711
LSF_EAUTH_UID.....	712
LSF_EXECUTE_DOMAIN.....	712
LSF_INTERACTIVE_STDERR.....	712
LSF_INVOKE_CMD.....	712
LSF_JOB_STARTER.....	713
LSF_LD_LIBRARY_PATH.....	714
LSF_LD_PRELOAD.....	714
LSF_LIM_API_NTRIES.....	714
LSF_LIM_DEBUG.....	715
LSF_LOGDIR.....	715
LSF_LSLOAD_FORMAT.....	715
LSF_LSHOSTS_FORMAT.....	715

LSF_MASTER.....	715
LSF_NIOS_DEBUG.....	715
LSF_NIOS_ERR_LOGDIR.....	715
LSF_NIOS_DIE_CMD.....	716
LSF_NIOS_IGNORE_SIGWINDOW.....	716
LSF_NIOS_PEND_TIMEOUT.....	716
LSF_NIOS_PORT_RANGE.....	717
LSF_RESOURCES.....	717
LSF_TS_LOGON_TIME	717
LSF_USE_HOSTEQUIV	718
LSF_USER_DOMAIN.....	718
NOCHECKVIEW_POSTEXEC.....	719
OMP_NUM_THREADS.....	719
RM_CPUTASK _n	720
RM_MEM_AFFINITY.....	721
TASKMAN_EXEC_RUSAGE.....	721

Chapter 1. Configuration files

LSF configuration files reference.

Important:

Specify any domain names in all uppercase letters in all configuration files.

cshrc.lsf and profile.lsf

The user environment shell files `cshrc.lsf` and `profile.lsf` set the LSF operating environment on an LSF host.

The `cshrc.lsf` and `profile.lsf` files define machine-dependent paths to LSF commands and libraries as environment variables:

- The `cshrc.lsf` file sets the C shell (**csh** or **tcsh**) user environment for LSF commands and libraries.
- The `profile.lsf` file sets and exports the Bourne shell/Korn shell (**sh**, **ksh**, or **bash**) user environment for LSF commands and libraries.

Tip: LSF Administrators can make sure that `cshrc.lsf` or `profile.lsf` are available for users to set the LSF environment variables correctly for the host type that is running LSF.

Location

The `cshrc.lsf` and `profile.lsf` files are created by the **lsfinstall** script during installation. After installation, they are located in the **LSF_CONFDIR** directory (`LSF_TOP/conf/`).

Format

The `cshrc.lsf` and `profile.lsf` files are conventional UNIX shell scripts:

- `cshrc.lsf` runs under **/bin/csh**
- `profile.lsf` runs under **/bin/sh**

What cshrc.lsf and profile.lsf do

The `cshrc.lsf` and `profile.lsf` files determine the binary type (**BINARY_TYPE**) of the host and set environment variables for the paths to the following machine-dependent LSF directories, according to the LSF version (**LSF_VERSION**) and the location of the top-level installation directory (**LSF_TOP**) defined at installation:

- **LSF_BINDIR**
- **LSF_SERVERDIR**
- **LSF_LIBDIR**

The `cshrc.lsf` and `profile.lsf` files also set the following user environment variables:

- **LSF_ENVDIR**
- **LD_LIBRARY_PATH**
- **PATH** to include the paths to the following directories:
 - **LSF_BINDIR**
 - **LSF_SERVERDIR**
- **MANPATH** to include the path to the LSF man pages

If enterprise grid orchestrator is enabled

If enterprise grid orchestrator is enabled in the LSF cluster (the **LSF_ENABLE_EGO=Y** and **LSF_EGO_ENVDIR** parameters are defined in the `lsf.conf` file), the `cshrc.lsf` and `profile.lsf` files set the following environment variables.

- **EGO_BINDIR**
- **EGO_CONFDIR**
- **EGO_ESRVDIR**
- **EGO_LIBDIR**
- **EGO_LOCAL_CONFDIR**
- **EGO_SERVERDIR**
- **EGO_TOP**

Set the LSF environment with `cshrc.lsf` and `profile.lsf`

Before you use LSF, you must use the `cshrc.lsf` or `profile.lsf` file to set the LSF execution environment.

Log in to an LSF host and use one of the following shell environment files to set your LSF environment:

- In the **csh** or **tcsh** shell, run the **source** command:

```
% source <LSF_TOP>/conf/cshrc.lsf
```

- In the **sh**, **ksh**, or **bash** shell, run the following command:

```
$ . <LSF_TOP>/conf/profile.lsf
```

Making your cluster available to users with `cshrc.lsf` and `profile.lsf`

Tip: LSF administrators can make sure all LSF users include one of these files at the end of their own `.cshrc` or `.profile` file, or run one of these two files before you use LSF.

For `csh` or `tcsh`

Add the `cshrc.lsf` file to the end of the `.cshrc` file for all users:

- Copy the contents of the `cshrc.lsf` file into the `.cshrc` file.
- Add a line with the **source** command to the end of the `.cshrc` file.

For example, if the `LSF_TOP` directory for your cluster is `/usr/share/lsf/conf`, add the following line to the `.cshrc` file:

```
source /usr/share/lsf/conf/cshrc.lsf
```

After you run the `cshrc.lsf` script, use the **setenv** command to see the environment variable settings.

```
setenv
...
PATH=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/bin
MANPATH=/usr/lsf/llsf_10/10.1/man
LSF_BINDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/bin
LSF_SERVERDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/etc
LSF_LIBDIR=/usr/lsf/10.1/linux2.6-glibc2.3-x86/lib
LD_LIBRARY_PATH=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/lib
XLSF_UIDDIR=/usr/lsf/10.1/linux2.6-glibc2.3-x86/lib/uid
LSF_ENVDIR=/usr/lsf/lsf_10/conf
...
```

Note: These variable settings are only an example. Your system might set extra variables.

For sh, ksh, or bash

Add `profile.lsf` to the end of the `.profile` file for all users:

- Copy the contents of the `profile.lsf` file into the `.profile` file.
- Add a line that runs the `profile.lsf` file to the end of the `.profile` file.

For example, if the `LSF_TOP` directory for your cluster is `/usr/share/lsf/conf`, add the following line to the `.profile` file:

```
. /usr/share/lsf/conf/profile.lsf
```

After you run the `profile.lsf` file, use the `setenv` command to see the environment variable settings.

```
setenv
...
LD_LIBRARY_PATH=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/lib
LSF_BINDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/bin
LSF_ENVDIR=/usr/lsf/lsf_10/conf
LSF_LIBDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/lib
LSF_SERVERDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/etc
MANPATH=/usr/lsf/lsf_10/10.1/man
PATH=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/bin
XLSF_UIDDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/lib/uid
...
```

Note: These variable settings are only an example. Your system might set more variables.

cshrc.lsf and profile.lsf on dynamically added LSF hosts

Dynamically added LSF hosts that are not master candidates are called *slave hosts*. Each dynamic host has its own LSF binary files, local `lsf.conf` file, and shell environment scripts (`cshrc.lsf` and `profile.lsf`).

LSF environment variables set by cshrc.lsf and profile.lsf

The `cshrc.lsf` and `profile.lsf` files set several LSF environment variables.

LSF_BINDIR

The `LSF_BINDIR` environment variable is set by the `cshrc.lsf` and `profile.lsf` files.

Syntax

```
LSF_BINDIR=dir
```

Description

Directory where LSF user commands are installed.

Examples

- Set in **csh** and **tcsh** shells by the `cshrc.lsf` file:

```
setenv LSF_BINDIR /usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/bin
```

- Set and exported in the **sh**, **ksh**, or **bash** shell by the `profile.lsf` file:

```
LSF_BINDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/bin
```

Values

- In the `cshrc.lsf` file for the **csh** and **tcsh** shells:

```
setenv LSF_BINDIR $LSF_TOP/$LSF_VERSION/$BINARY_TYPE/bin
```

- Set and exported in the profile.lsf file for the **sh**, **ksh**, or **bash** shell:

```
LSF_BINDIR=$LSF_TOP/$LSF_VERSION/$BINARY_TYPE/bin
```

LSF_ENVDIR

Directory containing the lsf.conf file.

Syntax

```
LSF_ENVDIR=dir
```

Description

By default, the lsf.conf file is installed by creating a shared copy in the **LSF_CONFDIR** directory and adding a symbolic link from the /etc/lsf.conf directory to the shared copy. If the **LSF_ENVDIR** variable is set, the symbolic link is installed in the LSF_ENVDIR/lsf.conf file.

The lsf.conf file is a global environment configuration file for all LSF services and applications. The LSF default installation places the file in the **LSF_CONFDIR** directory.

Examples

- Set in the **cskr** and **tcscr** shells by the cskrc.lsf file:

```
setenv LSF_ENVDIR /usr/lsf/lsf_10/conf
```

- Set and exported in the **sh**, **ksh**, or **bash** shell by the profile.lsf file:

```
LSF_ENVDIR=/usr/lsf/lsf_10/conf
```

Values

- In the cskrc.lsf file for the **cskr** and **tcscr** shells:

```
setenv LSF_ENVDIR $LSF_TOP/conf
```

- Set and exported in the profile.lsf file for the **sh**, **ksh**, or **bash** shell:

```
LSF_ENVDIR=$LSF_TOP/conf
```

LSF_LIBDIR

Directory where LSF libraries are installed.

Syntax

```
LSF_LIBDIR=dir
```

Description

Library files are shared by all hosts of the same type.

Examples

- Set in the **cskr** and **tcscr** shells by the cskrc.lsf file:

```
setenv LSF_LIBDIR /usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/lib
```

- Set and exported in the **sh**, **ksh**, or **bash** shell by the profile.lsf file:

```
LSF_LIBDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/lib
```

Values

- In the `cshrc.lsf` file for the **csh** and **tcsh** shells:

```
setenv LSF_LIBDIR $LSF_TOP/$LSF_VERSION/$BINARY_TYPE/lib
```

- Set and exported in the `profile.lsf` files for the **sh**, **ksh**, or **bash** shell:

```
LSF_LIBDIR=$LSF_TOP/$LSF_VERSION/$BINARY_TYPE/lib
```

LSF_SERVERDIR

Directory where LSF server binary files and shell scripts are installed.

Syntax

`LSF_SERVERDIR=dir`

Description

These server binary files include the **lim**, **res**, **nios**, **sbatchd**, **mbatchd**, and **mbschd** daemons. If you use **elim**, **eauth**, **eexec**, **esub**, or other external executable files, they are also installed in this directory.

Examples

- Set in the **csh** and **tcsh** shells by `cshrc.lsf` files:

```
setenv LSF_SERVERDIR /usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/etc
```

- Set and exported in the **sh**, **ksh**, or **bash** shell by the `profile.lsf` file:

```
LSF_SERVERDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/etc
```

Values

- In the `cshrc.lsf` file for the **csh** and **tcsh** shells:

```
setenv LSF_SERVERDIR $LSF_TOP/$LSF_VERSION/$BINARY_TYPE/etc
```

- Set and exported in the `profile.lsf` file for the **sh**, **ksh**, or **bash** shell:

```
LSF_SERVERDIR=$LSF_TOP/$LSF_VERSION/$BINARY_TYPE/etc
```

EGO environment variables set by cshrc.lsf and profile.lsf

The `cshrc.lsf` and `profile.lsf` files set the following EGO environment variables: **EGO_BINDIR**, **EGO_CONFDIR**, **EGO_ESRVDIR**, **EGO_LIBDIR**, **EGO_LOCAL_CONFDIR**, **EGO_SERVERDIR**, **EGO_TOP**.

EGO_BINDIR

Directory where EGO user commands are installed.

Syntax

`EGO_BINDIR=dir`

Description

The **EGO_BINDIR** environment variable sets the directory where EGO user commands are installed.

Examples

- Set in the **cskr** and **tcscr** shells by the `cskr.lsf` file:

```
setenv EGO_BINDIR /usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/bin
```

- Set and exported in the **sh**, **ksh**, or **bash** shell by the `profile.lsf` file:

```
EGO_BINDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/bin
```

Values

- In the `cskr.lsf` file for the **cskr** and **tcscr** shells:

```
setenv EGO_BINDIR $LSF_BINDIR
```

- Set and exported in the `profile.lsf` file for the **sh**, **ksh**, or **bash** shell:

```
EGO_BINDIR=$LSF_BINDIR
```

EGO_CONFDIR

Directory containing the `ego.conf` file.

Syntax

`EGO_CONFDIR=dir`

Description

EGO_CONFDIR environment variable sets the directory that contains the `ego.conf` file.

Examples

- Set in the **cskr** and **tcscr** shells by the `cskr.lsf` file:

```
setenv EGO_CONFDIR /usr/lsf/lsf_10/conf/ego/lsf1.2.3/kernel
```

- Set and exported in the **sh**, **ksh**, or **bash** shell by the `profile.lsf` file:

```
EGO_CONFDIR=/usr/lsf/lsf_10/conf/ego/lsf1.2.3/kernel
```

Values

- In the `cskr.lsf` file for the **cskr** and **tcscr** shells:

```
setenv EGO_CONFDIR /usr/lsf/lsf_10/conf/ego/lsf1.2.3/kernel
```

- Set and exported in the `profile.lsf` file for the **sh**, **ksh**, or **bash** shell:

```
EGO_CONFDIR=/usr/lsf/lsf_10/conf/ego/lsf1.2.3/kernel
```

EGO_ESRVDIR

Directory where the EGO the service controller configuration files are stored.

Syntax

`EGO_ESRVDIR=dir`

Description

The **EGO_ESRVDIR** environment variable sets the directory where the EGO the service controller configuration files are stored.

Examples

- Set in the **csh** and **tcsh** shells by the `cshrc.lsf` file:

```
setenv EGO_ESRVDIR /usr/lsf/lsf_10/conf/ego/lsf/eservice
```

- Set and exported in the **sh**, **ksh**, or **bash** shell by the `profile.lsf` file:

```
EGO_ESRVDIR=/usr/lsf/lsf_10/conf/ego/lsf/eservice
```

Values

- In the `cshrc.lsf` file for the **csh** and **tcsh** shells:

```
setenv EGO_ESRVDIR /usr/lsf/lsf_10/conf/ego/lsf/eservice
```

- Set and exported in the `profile.lsf` file for the **sh**, **ksh**, or **bash** shell:

```
EGO_ESRVDIR=/usr/lsf/lsf_10/conf/ego/lsf/eservice
```

EGO_LIBDIR

Directory where EGO libraries are installed.

Syntax

`EGO_LIBDIR=dir`

Description

The **EGO_LIBDIR** environment variable sets the directory where EGO libraries are installed. Library files are shared by all hosts of the same type.

Examples

- Set in the **csh** and **tcsh** shells by the `cshrc.lsf` file:

```
setenv EGO_LIBDIR /usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/lib
```

- Set and exported in the **sh**, **ksh**, or **bash** shell by the `profile.lsf` file:

```
EGO_LIBDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/lib
```

Values

- In the `cshrc.lsf` file for the **csh** and **tcsh** shells:

```
setenv EGO_LIBDIR $LSF_LIBDIR
```

- Set and exported in the `profile.lsf` file for the **sh**, **ksh**, or **bash** shell:

```
EGO_LIBDIR=$LSF_LIBDIR
```

EGO_LOCAL_CONFDIR

The local EGO configuration directory that contains the `ego.conf` file.

Syntax

`EGO_LOCAL_CONFDIR=dir`

Description

The **EGO_LOCAL_CONFDIR** environment variable sets the local EGO configuration directory that contains the `ego.conf` file.

Examples

- Set in **csh** and **tcsh** by `cshrc.lsf`:

```
setenv EGO_LOCAL_CONFDIR /usr/lsf/llsf_10/conf/ego/lsf1.2.3/kernel
```

- Set and exported in **sh**, **ksh**, or **bash** by `profile.lsf`:

```
EGO_LOCAL_CONFDIR=/usr/lsf/lsf_10/conf/ego/lsf1.2.3/kernel
```

Values

- In `cshrc.lsf` for **csh** and **tcsh**:

```
setenv EGO_LOCAL_CONFDIR /usr/lsf/lsf_10/conf/ego/lsf1.2.3/kernel
```

- Set and exported in `profile.lsf` for **sh**, **ksh**, or **bash**:

```
EGO_LOCAL_CONFDIR=/usr/lsf/lsf_10/conf/ego/lsf1.2.3/kernel
```

EGO_SERVERDIR

Directory where EGO server binary files and shell scripts are installed.

Syntax

```
EGO_SERVERDIR=dir
```

Description

The server binary files in the **EGO_SERVERDIR** directory include **vemkd**, **pem**, **egosc**, and shell scripts for EGO startup and shutdown.

Examples

- Set in **csh** and **tcsh** by `cshrc.lsf`:

```
setenv EGO_SERVERDIR /usr/lsf/llsf_10/10.1/linux2.6-glibc2.3-x86/etc
```

- Set and exported in **sh**, **ksh**, or **bash** by `profile.lsf`:

```
EGO_SERVERDIR=/usr/lsf/lsf_10/10.1/linux2.6-glibc2.3-x86/etc
```

Values

- In `cshrc.lsf` for **csh** and **tcsh**:

```
setenv EGO_SERVERDIR $LSF_SERVERDIR
```

- Set and exported in `profile.lsf` for **sh**, **ksh**, or **bash**:

```
EGO_SERVERDIR=$LSF_SERVERDIR
```

EGO_TOP

The top-level installation directory for EGO.

Syntax

EGO_TOP=*dir*

The path to the **EGO_TOP** directory must be shared and accessible to all hosts in the cluster. Equivalent to the **LSF_TOP** directory.

Examples

- Set in **csh** and **tcsh** by `cshrc.lsf`:

```
setenv EGO_TOP /usr/lsf/lsf_10
```

- Set and exported in **sh**, **ksh**, or **bash** by `profile.lsf`:

```
EGO_TOP=/usr/lsf/lsf_10
```

Values

- In `cshrc.lsf` for **csh** and **tcsh**:

```
setenv EGO_TOP /usr/lsf/lsf_10
```

- Set and exported in `profile.lsf` for **sh**, **ksh**, or **bash**:

```
EGO_TOP=/usr/lsf/lsf_10
```

hosts

For hosts with multiple IP addresses and different official host names configured at the system level, this file associates the host names and IP addresses in LSF.

By default, LSF assumes each host in the cluster has the following properties:

- Has a unique official host name
- Can resolve its IP address from its name
- Can resolve its official name from its IP address

Hosts with only one IP address, or hosts with multiple IP addresses that already resolve to a unique official host name should not be configured in this file: they are resolved using the default method for your system (for example, local configuration files like `/etc/hosts` or through DNS.)

The LSF `hosts` file is used in the following environments:

- Machines in cluster have multiple network interfaces and cannot be set up in the system with a unique official host name
- DNS is slow or not configured properly
- Machines have special topology requirements; for example, in HPC systems where it is desirable to map multiple actual hosts to a single head end host

The LSF `hosts` file is not installed by default. It is usually located in the directory specified by the **LSF_CONFDIR** parameter. The format of the `LSF_CONFDIR/hosts` file is similar to the format of the `/etc/hosts` file on UNIX machines.

hosts file structure

One line for each IP address, consisting of the IP address, followed by the official host name, optionally followed by host aliases, all separated by spaces or tabs. Each line has the form:

```
ip_address official_name [alias [alias ...]]
```

IP addresses can have either a dotted quad notation (IPv4) or IP Next Generation (IPv6) format. You can use IPv6 addresses if you define the parameter **LSF_ENABLE_SUPPORT_IPV6** in the `lsf.conf` file. You do not have to map IPv4 addresses to an IPv6 format.

Use consecutive lines for IP addresses belonging to the same host. You can assign different aliases to different addresses.

Use a pound sign (`#`) to indicate a comment (the rest of the line is not read by LSF). Do not use `#if` because it is reserved syntax for time-based configuration.

IP address

Written using an IPv4 or IPv6 format. LSF supports both formats; you do not have to map IPv4 addresses to an IPv6 format (if you define the parameter **LSF_ENABLE_SUPPORT_IPV6** in the `lsf.conf` file).

- IPv4 format: `nnn.nnn.nnn.nnn`
- IPv6 format: `nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn`

Official host name

The official host name. Single character names are not allowed.

Specify `-GATEWAY` or `-GW` as part of the host name if the host serves as a GATEWAY.

Specify `-TAC` as the last part of the host name if the host is a TAC and is a DoD host.

Specify the host name in the format defined in Internet RFC 952, which states:

A name (Net, Host, Gateway, or Domain name) is a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9), minus sign (-), and period (.). Periods are only allowed when they serve to delimit components of domain style names. (See RFC 921, Domain Name System Implementation Schedule, for background). No blank or space characters are permitted as part of a name. No distinction is made between upper and lower case. The first character must be an alpha character. The last character must not be a minus sign or a period.

RFC 952 has been modified by RFC 1123 to relax the restriction on the first character being a digit.

For maximum interoperability with the Internet, you should use host names no longer than 24 characters for the host portion (exclusive of the domain component).

Aliases

Optional. Aliases to the host name.

The default host file syntax

```
ip_address official_name [alias [alias ...]]
```

is powerful and flexible, but it is difficult to configure in systems where a single host name has many aliases, and in multihomed host environments.

In these cases, the `hosts` file can become very large and unmanageable, and configuration is prone to error.

The syntax of the LSF `hosts` file supports host name ranges as aliases for an IP address. This simplifies the host name alias specification.

To use host name ranges as aliases, the host names must consist of a fixed node group name prefix and node indices, specified in a form like:

```
host_name[index_x-index_y, index_m, index_a-index_b]
```

For example:

```
atlasD0[0-3,4,5-6, ...]
```

is equivalent to:

```
atlasD0[0-6, ...]
```

The node list does not need to be a continuous range (some nodes can be configured out). Node indices can be numbers or letters (both upper case and lower case).

For example, some systems map internal compute nodes to single LSF host names. A host file might contain 64 lines, each specifying an LSF host name and 32 node names that correspond to each LSF host:

```
...
177.16.1.1 atlasD0 atlas0 atlas1 atlas2 atlas3 atlas4 ... atlas31
177.16.1.2 atlasD1 atlas32 atlas33 atlas34 atlas35 atlas36 ... atlas63
...
```

In the new format, you still map the nodes to the LSF hosts, so the number of lines remains the same, but the format is simplified because you only have to specify ranges for the nodes, not each node individually as an alias:

```
...
177.16.1.1 atlasD0 atlas[0-31]
177.16.1.2 atlasD1 atlas[32-63]
...
```

You can use either an IPv4 or an IPv6 format for the IP address (if you define the parameter `LSF_ENABLE_SUPPORT_IPV6` in `lsf.conf`).

IPv4 Example

```
192.168.1.1 hostA hostB
192.168.2.2 hostA hostC host-C
```

In this example, `hostA` has 2 IP addresses and 3 aliases. The alias `hostB` specifies the first address, and the aliases `hostC` and `host-C` specify the second address. LSF uses the official host name, `hostA`, to identify that both IP addresses belong to the same host.

IPv6 Example

```
3ffe:b80:3:1a91::2 hostA hostB 3ffe:b80:3:1a91::3 hostA hostC host-C
```

In this example, `hostA` has 2 IP addresses and 3 aliases. The alias `hostB` specifies the first address, and the aliases `hostC` and `host-C` specify the second address. LSF uses the official host name, `hostA`, to identify that both IP addresses belong to the same host.

install.config

The `install.config` file contains options for LSF installation and configuration. Use the **lsfinstall -f install.config** command to install LSF with the options that are specified in the `install.config` file.

Template location

A template `install.config` is included in the installer script package `lsf10.1.0.9_lsfinstall.tar.Z` and is located in the `lsf10.1.0.9_lsfinstall` directory that is created when you decompress and extract the installer script package. Edit the file and uncomment the options that you want in the template file. Replace the example values with your own settings to specify the options for your new installation.

Important:

The sample values in the `install.config` template file are examples only. They are not default installation values.

After installation, the `install.config` file that contains the options that you specified is located in the `LSF_TOP/10.1/install/` directory.

Format

Each entry in the `install.config` file has the following form:

```
NAME="STRING1 STRING2 ..."
```

The equal sign `=` must follow each NAME parameter even if no value follows and there must be no spaces around the equal sign.

A value that contains multiple strings that are separated by spaces must be enclosed in quotation marks.

Blank lines and lines that start with a number sign (`#`) are ignored.

Parameters

- CONFIGURATION_TEMPLATE
- EGO_DAEMON_CONTROL
- ENABLE_CGROUP
- ENABLE_DYNAMIC_HOSTS
- ENABLE_EGO
- ENABLE_STREAM
- LSF_ADD_SERVERS
- LSF_ADD_CLIENTS
- LSF_ADMINS
- LSF_CLUSTER_NAME
- LSF_DYNAMIC_HOST_WAIT_TIME
- LSF_ENTITLEMENT_FILE
- LSF_MASTER_LIST
- LSF_QUIET_INST
- LSF_SILENT_INSTALL_TARLIST
- LSF_TARDIR
- LSF_TOP
- PATCH_BACKUP_DIR

- PATCH_HISTORY_DIR
- SILENT_INSTALL

CONFIGURATION_TEMPLATE

Syntax

CONFIGURATION_TEMPLATE="DEFAULT" | "PARALLEL" | "HIGH_THROUGHPUT"

Description

LSF Standard Edition on UNIX or Linux only. Selects the configuration template for this installation, which determines the initial LSF configuration parameters that are specified when the installation is complete. The following are valid values for this parameter:

DEFAULT

Use this template for clusters with mixed workload. This configuration can serve different types of workload with good performance, but is not tuned for a particular type of cluster.

PARALLEL

This template provides extra support for large parallel jobs. Since this configuration is designed for long running parallel jobs, do not use this configuration for clusters that mainly run short jobs due to the longer reporting time for each job.

HIGH_THROUGHPUT

This template is used for clusters that mainly run short jobs, where over 80% of jobs finish within one minute. This high turnover rate requires LSF to be more responsive and fast acting. However, this configuration uses more resources as the daemons become busier.

The installer uses the DEFAULT configuration template when installing LSF Standard Edition on Windows

Note: Do not specify the **CONFIGURATION_TEMPLATE** parameter for LSF Express Edition and Advanced Edition. These editions have their own default configuration templates for all installations.

The installer specifies the following initial configuration file parameter values based on the selected configuration template:

- DEFAULT

lsf.conf

```
DAEMON_SHUTDOWN_DELAY=180
LSF_LINUX_CGROUP_ACCT=Y
LSF_PROCESS_TRACKING=Y
```

lsb.params

```
JOB_DEP_LAST_SUB=1
JOB_SCHEDULING_INTERVAL=1
MAX_JOB_NUM=10000
NEWJOB_REFRESH=Y
SBD_SLEEP_TIME=7
```

- PARALLEL

lsf.conf

```
LSB_SHORT_HOSTLIST=1
LSF_LINUX_CGROUP_ACCT=Y
LSF_PROCESS_TRACKING=Y
LSF_ENABLE_EXTSCHEDULER=Y
LSF_HPC_EXTENSIONS="CUMULATIVE_RUSAGE LSB_HCLOSE_BY_RES SHORT_EVENTFILE"
```

For a full description of the PARALLEL configuration template, refer to *Enable LSF HPC features in Installing IBM Spectrum LSF on UNIX and Linux*.

lsb.params

```
JOB_DEP_LAST_SUB=1
JOB_SCHEDULING_INTERVAL=1
NEWJOB_REFRESH=Y
TRACK_ELIGIBLE_PENDINFO=Y
```

- HIGH_THROUGHPUT

lsf.conf

```
LSB_MAX_PACK_JOBS=300
LSB_SHORT_HOSTLIST=1
```

lsb.params

```
JOB_SCHEDULING_INTERVAL=50ms
MAX_INFO_DIRS=500
MAX_JOB_ARRAY_SIZE=10000
MAX_JOB_NUM=100000
MIN_SWITCH_PERIOD=1800
NEWJOB_REFRESH=YSBD_SLEEP_TIME=3
```

The installer specifies the following initial configuration parameters for all configuration templates:

- **lsf.conf:**

```
EGO_ENABLE_AUTO_DAEMON_SHUTDOWN=Y
LSB_DISABLE_LIMLOCK_EXCL=Y
LSB_MOD_ALL_JOBS=Y
LSF_DISABLE_LSRUN=Y
LSB_SUBK_SHOW_EXEC_HOST=Y
LSF_PIM_LINUX_ENHANCE=Y
LSF_PIM_SLEEP_TIME_UPDATE=Y
LSF_STRICT_RESREQ=Y
LSF_UNIT_FOR_LIMITS=MB
```

- **lsb.params:**

```
ABS_RUNLIMIT=Y
DEFAULT_QUEUE=normal interactive
JOB_ACCEPT_INTERVAL=0
MAX_CONCURRENT_QUERY=100
MAX_JOB_NUM=10000
MBD_SLEEP_TIME=10
PARALLEL_SCHED_BY_SLOT=Y
RELAX_JOB_DISPATCH_ORDER=Y
```

In addition, the installer enables the following features for all configuration templates:

- Fairshare scheduling (LSF Standard Edition and Advanced Edition): The following example for the `lsb.queues` file enables fairshare scheduling for all queues except `admin` and `license`:

```
Begin Queue
...
FAIRSHARE=USER_SHARES[[default, 1]]
...
End Queue
```

- Host groups (LSF Standard Edition on UNIX or Linux): Master candidate hosts are assigned to the `master_hosts` host group.
- User groups (LSF Standard Edition on UNIX or Linux): LSF administrators are assigned to the `lsfadmins` user group.
- Affinity scheduling in both the `lsb.modules` and `lsb.hosts`.

Example

```
CONFIGURATION_TEMPLATE="HIGH_THROUGHPUT"
```

Default

DEFAULT (the default configuration template is used)

EGO_DAEMON_CONTROL**Syntax**

```
EGO_DAEMON_CONTROL="Y" | "N"
```

Description

Enables EGO to control the LSF **res** and **sbatchd** daemons. Set the value to "Y" if you want the EGO service controller to start the **res** and **sbatchd** daemons, and restart if they fail. To avoid conflicts, leave this parameter undefined if you use a script to start LSF daemons.

Note: If you specify EGO_ENABLE="N", this parameter is ignored.

Example

```
EGO_DAEMON_CONTROL="N"
```

Default

N (the **res** and **sbatchd** are started manually)

ENABLE_CGROUP**Syntax**

```
ENABLE_CGROUP=Y | N
```

Description

Enables LSF to track processes' CPU and memory accounting based on Linux **cgroup** memory and **cpuacct** subsystems.

Set the value to Y if you want to use Linux **cgroup** to track process accounting. The installer sets the following initial configuration file parameter values in the `lsf.conf` file:

```
LSB_RESOURCE_ENFORCE="cpu memory"
LSF_PROCESS_TRACKING=Y
LSF_LINUX_CGROUP_ACCT=Y
```

Example

```
ENABLE_CGROUP=Y
```

Default

N (do not use Linux **cgroup** to track process accounting)

ENABLE_DYNAMIC_HOSTS**Syntax**

```
ENABLE_DYNAMIC_HOSTS="Y" | "N"
```

Description

Enables dynamically adding and removing hosts. Set the value to "Y" if you want to allow dynamically added hosts.

If you enable dynamic hosts, any host can connect to cluster. To enable security, configure the **LSF_HOST_ADDR_RANGE** parameter in the `lsf.cluster.cluster_name` file after installation and restrict the hosts that can connect to your cluster.

Example

```
ENABLE_DYNAMIC_HOSTS="N"
```

Default

N (dynamic hosts not allowed)

ENABLE_EGO**Syntax**

```
ENABLE_EGO="Y" | "N"
```

Description

Enables EGO functions in the LSF cluster.

ENABLE_EGO="Y" causes the **lsfinstall** command to uncomment the **LSF_EGO_ENVDIR** parameter and sets the **LSF_ENABLE_EGO="Y"** parameter in the `lsf.conf` file.

ENABLE_EGO="N" causes the **lsfinstall** command to comment out the **LSF_EGO_ENVDIR** parameter and sets the **LSF_ENABLE_EGO="N"** parameter in the `lsf.conf` file.

Set the value to **ENABLE_EGO="Y"** if you want to take advantage of the following LSF features that depend on EGO:

- LSF daemon control by EGO service controller
- SLA scheduling with EGO enabled

Default

N (EGO is disabled in the LSF cluster)

ENABLE_GPU**Syntax**

```
ENABLE_GPU=Y | N
```

Description

Enables LSF to support GPUs so that applications can use GPU resources in a Linux environment. LSF supports parallel jobs that require GPUs based on availability.

Set the value to Y if you want to run parallel jobs that request GPU resources in Linux environments. The installer sets the following initial configuration file parameter values in the following configuration files:

- `lsf.conf`:

```
LSB_RESOURCE_ENFORCE="gpu"
```

- `lsf.cluster.cluster_name`:

```
Begin ResourceMap
RESOURCENAME LOCATION
ngpus          ([default])
ngpus_shared   ([default])
ngpus_excl_t   ([default])
ngpus_excl_p   ([default])
...
```

```
gpu_topology      ([default])
End ResourceMap
```

There are also several resources that are added to the file, but are commented out by default. The resources that are commented out are not shown here.

- `lsf.shared`:

```
Begin Resource
RESOURCENAME  TYPE      INTERVAL  INCREASING  CONSUMABLE  DESCRIPTION  # Keywords
  ngpus        Numeric    60        N           N           N           (Number of GPUs)
  ngpus_shared Numeric    60        N           N           Y           (Number of GPUs in Shared
Mode)
  ngpus_excl_t Numeric    60        N           Y           Y           (Number of GPUs in
Exclusive Thread Mode)
  ngpus_excl_p Numeric    60        N           Y           Y           (Number of GPUs in
Exclusive Process Mode)
  ...
  gpu_topology String     60        ()          ()          ()          (GPU topology on host)
End Resource
```

There are also several resources that are added to the file, but are commented out by default. The resources that are commented out are not shown here.

Example

```
ENABLE_GPU=Y
```

Default

N (do not support GPUs)

ENABLE_STREAM

Syntax

```
ENABLE_STREAM="Y" | "N"
```

Description

Enables LSF event streaming.

Enable LSF event streaming if you intend to install IBM Spectrum LSF Analytics or IBM Spectrum LSF Application Center.

Default

N (Event streaming is disabled)

LSF_ADD_SERVERS

Syntax

```
LSF_ADD_SERVERS="host_name [host_name...]"
```

Description

List of extra LSF server hosts.

The hosts in the **LSF_MASTER_LIST** parameter are always LSF servers. Use the **LSF_ADD_SERVERS** parameter to specify extra server hosts. Specify a list of host names two ways:

- Host names that are separated by spaces
- The name of a file that contains a list of host names, one host per line.

Valid Values

Any valid LSF host name.

Example 1

List of host names:

```
LSF_ADD_SERVERS="hosta hostb hostc hostd"
```

Example 2

Host list file:

```
LSF_ADD_SERVERS=:lsf_server_hosts
```

The `lsf_server_hosts` file contains a list of hosts:

```
hosta  
hostb  
hostc  
hostd
```

Default

Only hosts in the **LSF_MASTER_LIST** parameter are LSF servers.

LSF_ADD_CLIENTS

Syntax

```
LSF_ADD_CLIENTS="host_name [host_name...]"
```

Description

List of LSF client-only hosts.

Tip:

After installation, you must manually edit the `lsf.cluster.cluster_name` file to include the host model and type of each client that is listed in the **LSF_ADD_CLIENTS** parameter.

Valid Values

Any valid LSF host name.

Example 1

List of host names:

```
LSF_ADD_CLIENTS="hoste hostf"
```

Example 2

Host list file:

```
LSF_ADD_CLIENTS=:lsf_client_hosts
```

The `lsf_client_hosts` file contains a list of hosts:

```
hoste  
hostf
```

Default

No client hosts installed.

LSF_ADMINS**Syntax**

```
LSF_ADMINS="user_name [user_name ... ]"
```

Description

Required. List of LSF administrators.

The first user account name in the list is the primary LSF administrator. This user name cannot be the root user account.

Typically, this account is named `lsfadmin`. This account owns the LSF configuration files and log files for job events. This account also has permission to reconfigure LSF and to control batch jobs that are submitted by other users. The primary LSF administrator typically does not have authority to start LSF daemons. Usually, only root has permission to start LSF daemons.

All the LSF administrator accounts must exist on all hosts in the cluster before you install LSF. Secondary LSF administrators are optional.



CAUTION: You cannot configure the root account as the primary LSF administrator.

Valid Values

Existing user accounts

Example

```
LSF_ADMINS="lsfadmin user1 user2"
```

Default

None - required variable

LSF_CLUSTER_NAME**Syntax**

```
LSF_CLUSTER_NAME="cluster_name"
```

Description

Required. The name of the LSF cluster.

Example

```
LSF_CLUSTER_NAME="cluster1"
```

Valid Values

Any alphanumeric string that contains no more than 39 characters. The name cannot contain white spaces.

Important:

Do not use the name of any host, user, or user group as the name of your cluster.

Default

None - required variable

LSF_DYNAMIC_HOST_WAIT_TIME

Syntax

LSF_DYNAMIC_HOST_WAIT_TIME=*seconds*

Description

Time in seconds that the slave LIM waits after startup before it calls the master LIM to add the slave host dynamically.

This parameter takes effect only if you set **ENABLE_DYNAMIC_HOSTS="Y"** in this file. If the slave LIM receives the master announcement while it is waiting, it does not call the master LIM to add itself.

Specify a value up to 60 seconds for every 1000 hosts in the cluster, for a maximum of 15 minutes. Selecting a smaller value will result in a quicker response time for new hosts at the expense of an increased load on the master LIM.

Example

LSF_DYNAMIC_HOST_WAIT_TIME=60

Hosts will wait 60 seconds from startup to receive an acknowledgment from the master LIM. If it does not receive the acknowledgment within the 60 seconds, it will send a request for the master LIM to add it to the cluster.

Default

Slave LIM waits forever

LSF_ENTITLEMENT_FILE

Syntax

LSF_ENTITLEMENT_FILE=*path*

Description

Full path to the LSF entitlement file. LSF uses the entitlement to determine which feature set to enable or disable based on the edition of the product. The entitlement file for LSF Standard Edition is `lsf_std_entitlement.dat`. For LSF Express Edition, the file is `lsf_exp_entitlement.dat`. For LSF Advanced Edition, the file is `lsf_adv_entitlement.dat`. The entitlement file is installed as `<LSF_TOP>/conf/lsf.entitlement`.

You must download the entitlement file for the edition of the product you are running, and set the **LSF_ENTITLEMENT_FILE** parameter to the full path to the entitlement file you downloaded.

After LSF is installed and running, run the **lsid** command to see which edition of LSF is enabled.

Example

LSF_ENTITLEMENT_FILE=/usr/share/lsf_distrib/lsf.entitlement

Default

None - required variable

LSF_MASTER_LIST**Syntax**

```
LSF_MASTER_LIST="host_name [host_name ...]"
```

Description

Required for a first-time installation. List of LSF server hosts to be master or master candidates in the cluster.

You must specify at least one valid server host to start the cluster. The first host that is listed is the LSF master host.

During upgrade, specify the existing value.

Valid Values

LSF server host names

Example

```
LSF_MASTER_LIST="hosta hostb hostc hostd"
```

Default

None - required variable

LSF_QUIET_INST**Syntax**

```
LSF_QUIET_INST="Y" | "N"
```

Description

Enables quiet installation.

Set the value to Y if you want to hide the LSF installation messages.

Example

```
LSF_QUIET_INST="Y"
```

Default

N (installer displays messages during installation)

LSF_SILENT_INSTALL_TARLIST**Syntax**

```
LSF_SILENT_INSTALL_TARLIST="ALL" | "Package_Name ..."
```

Description**Example**

A string that contains all LSF package names to be installed. This name list applies only to the silent installation mode. Supports keywords `all`, `ALL`, and `All`, which can install all packages in the directory that is specified by the **LSF_TARDIR** parameter.

LSF_SILENT_INSTALL_TARLIST="ALL" | "lsf10.1.0.9_linux2.6-glibc2.3-x86_64.tar.Z"

Default

None

LSF_TARDIR

Syntax

LSF_TARDIR="/*path*"

Description

Full path to the directory that contains the LSF distribution TAR files.

Example

LSF_TARDIR="/usr/share/lsf_distrib"

Default

The parent directory of the current working directory. For example, if the **lsfinstall** command is running under the `usr/share/lsf_distrib/lsf_lsfinstall` directory, the default value of the **LSF_TARDIR** parameter is `usr/share/lsf_distrib`.

LSF_TOP

Syntax

LSF_TOP="/*path*"

Description

Required. Full path to the top level LSF installation directory.

Valid Value

The path to the **LSF_TOP** directory must be shared and accessible to all hosts in the cluster. It cannot be the root directory (`/`). The file system that contains the **LSF_TOP** directory must have enough disk space for all host types (approximately 300 MB per host type).

Example

LSF_TOP="/usr/share/lsf"

Default

None - required variable

PATCH_BACKUP_DIR

Syntax

PATCH_BACKUP_DIR="/*path*"

Description

Full path to the patch backup directory. This parameter is used when you install a new cluster for the first time, and is ignored for all other cases.

The file system that contains the patch backup directory must have sufficient disk space to back up your files, which is approximately 400 MB per binary type if you want to be able to install and roll back one enhancement pack and a few more fixes. It cannot be the root directory (/).

If the directory exists, it must be writable by the cluster administrator (lsfadmin).

If you need to change the directory after installation, edit the **PATCH_BACKUP_DIR** parameter in the `LSF_TOP/patch.conf` directory and move the saved backup files to the new directory manually.

Example

```
PATCH_BACKUP_DIR="/usr/share/lsf/patch/backup"
```

Default

```
LSF_TOP/patch/backup
```

PATCH_HISTORY_DIR

Syntax

```
PATCH_HISTORY_DIR="/path"
```

Description

Full path to the patch history directory. This parameter is used when you install a new cluster for the first time, and is ignored for all other cases.

It cannot be the root directory (/). If the directory exists, it must be writable by lsfadmin.

The location is saved as the **PATCH_HISTORY_DIR** parameter in the `LSF_TOP/patch.conf` directory. Do not change the directory after installation.

Example

```
PATCH_BACKUP_DIR="/usr/share/lsf/patch"
```

Default

```
LSF_TOP/patch
```

SILENT_INSTALL

Syntax

```
SILENT_INSTALL="Y" | "N"
```

Description

Enabling the silent installation (setting this parameter to Y) means that you want to do the silent installation and accept the license agreement.

Default

```
N
```

lim.acct

The `lim.acct` file is the log file for the LSF Load Information Manager (LIM). Produced by the `lsmon` command, the `lim.acct` file contains host load information that is collected and distributed by LIM.

lim.acct structure

The first line of `lim.acct` contains a list of load index names that are separated by spaces. This list of load index names can be specified in the `lsmon` command line. The default list is `r15s r1m r15m ut pg ls it swp mem tmp`. Subsequent lines in the file contain the host's load information at the time the information was recorded.

Fields

Fields are ordered in the following sequence:

time (%ld)

The time when the load information is written to the log file.

host name (%s)

The name of the host.

status of host (%d)

An array of integers. The first integer marks the operation status of the host. Remaining integers are used as a bitmap to indicate load status of the host. An integer can be used for 32 load indices. If the number of user-defined load indices is not more than 21, only one integer is used for both built-in load indices and external load indices. See the `hostload` structure in `ls_load(3)` for the description of these fields.

indexvalue (%f)

A sequence of load index values. Each value corresponds to the index name in the first line of `lim.acct`. The order in which the index values are listed is the same as the order of the index names.

lsb.acct

The `lsb.acct` file is the batch job log file of LSF.

The master batch daemon (`mbatchd`) generates a record for each job completion or failure. The record is appended to the job log file `lsb.acct`.

The file is located in the `LSB_SHAREDIR/cluster_name/logdir` directory, where the `LSB_SHAREDIR` parameter must be defined in `lsf.conf` and `cluster_name` is the name of the LSF cluster, as returned by `lsid`.

The `bacct` command uses the current `lsb.acct` file for its output.

lsb.acct structure

The job log file is an ASCII file with one record per line. The fields of a record are separated by blanks. If the value of some field is unavailable, a pair of double quotation marks ("") is logged for character string, 0 for time and number, and -1 for resource usage.

Configuring automatic archiving

The following parameters in `lsb.params` affect how records are logged to `lsb.acct`:

ACCT_ARCHIVE_AGE=*days*

Enables automatic archiving of LSF accounting log files, and specifies the archive interval. LSF archives the current log file if the length of time from its creation date exceeds the specified number of days.

By default there is no limit to the age of `lsb.acct`.

ACCT_ARCHIVE_SIZE=*kilobytes*

Enables automatic archiving of LSF accounting log files, and specifies the archive threshold. LSF archives the current log file if its size exceeds the specified number of kilobytes.

By default, there is no limit to the size of `lsb.acct`.

ACCT_ARCHIVE_TIME=*hh:mm*

Enables automatic archiving of LSF accounting log file `lsb.acct`, and specifies the time of day to archive the current log file.

By default, no time is set for archiving `lsb.acct`.

MAX_ACCT_ARCHIVE_FILE=*integer*

Enables automatic deletion of archived LSF accounting log files and specifies the archive limit.

By default, `lsb.acct.n` files are not automatically deleted.

Records and fields

The fields of a record are separated by blanks. The first string of an event record indicates its type. The following types of events are recorded:

- JOB_FINISH
- EVENT_ADRSV_FINISH
- JOB_RESIZE

JOB_FINISH

A job has finished.

If `LSF_HPC_EXTENSIONS="SHORT_EVENTFILE"` is specified in `lsf.conf`, older daemons and commands (pre-LSF Version 6.0) cannot recognize the `lsb.acct` file format.

The fields in order of occurrence are:

Event type (%s)

Which is JOB_FINISH

Version Number (%s)

Version number of the log file format

Event Time (%d)

Time the event was logged (in seconds since the epoch)

jobId (%d)

ID for the job

userId (%d)

UNIX user ID of the submitter

options (%d)

Bit flags for job processing

numProcessors (%d)

Number of processors initially requested for execution

submitTime (%d)

Job submission time

beginTime (%d)

Job start time – the job should be started at or after this time

termTime (%d)

Job termination deadline – the job should be terminated by this time

startTime (%d)

Job dispatch time – time job was dispatched for execution

userName (%s)

User name of the submitter

queue (%s)

Name of the job queue to which the job was submitted

resReq (%s)

Resource requirement specified by the user

dependCond (%s)

Job dependency condition specified by the user

preExecCmd (%s)

Pre-execution command specified by the user

fromHost (%s)

Submission host name

cwd (%s)

Current working directory (up to 4094 characters for UNIX or 512 characters for Windows), or the current working directory specified by **bsub -cwd** if that command was used.

inFile (%s)

Input file name (up to 4094 characters for UNIX or 512 characters for Windows)

outFile (%s)

output file name (up to 4094 characters for UNIX or 512 characters for Windows)

errFile (%s)

Error output file name (up to 4094 characters for UNIX or 512 characters for Windows)

jobFile (%s)

Job script file name

numAskedHosts (%d)

Number of host names to which job dispatching will be limited

askedHosts (%s)

List of host names to which job dispatching will be limited (%s for each); nothing is logged to the record for this value if the last field value is 0. If there is more than one host name, then each additional host name will be returned in its own field

numExHosts (%d)

Number of processors used for execution

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of hosts listed in the `execHosts` field.

Logged value reflects the allocation at job finish time.

execHosts (%s)

List of execution host names (%s for each); nothing is logged to the record for this value if the last field value is 0.

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format.

The logged value reflects the allocation at job finish time.

jStatus (%d)

Job status. The number 32 represents EXIT, 64 represents DONE

hostFactor (%f)

CPU factor of the first execution host.

jobName (%s)

Job name (up to 4094 characters).

command (%s)

Complete batch job command specified by the user (up to 4094 characters for UNIX or 512 characters for Windows).

lsfRusage

The following fields contain resource usage information for the job (see **getrusage(2)**). If the value of some field is unavailable (due to job exit or the difference among the operating systems), -1 will be logged. Times are measured in seconds, and sizes are measured in KB.

ru_utime (%f)

User time used

ru_stime (%f)

System time used

ru_maxrss (%f)

Maximum shared text size

ru_ixrss (%f)

Integral of the shared text size over time (in KB seconds)

ru_ismrss (%f)

Integral of the shared memory size over time (valid only on Ultrix)

ru_idrss (%f)

Integral of the unshared data size over time

ru_isrss (%f)

Integral of the unshared stack size over time

ru_minflt (%f)

Number of page reclaims

ru_majflt (%f)

Number of page faults

ru_nswap (%f)

Number of times the process was swapped out

ru_inblock (%f)

Number of block input operations

ru_oublock (%f)

Number of block output operations

ru_ioch (%f)

Number of characters read and written (valid only on HP-UX)

ru_msgsnd (%f)

Number of System V IPC messages sent

ru_msgrcv (%f)

Number of messages received

ru_nsignals (%f)

Number of signals received

ru_nvcsw (%f)

Number of voluntary context switches

ru_nivcsw (%f)

Number of involuntary context switches

ru_exutime (%f)

Exact user time used (valid only on ConvexOS)

mailUser (%s)

Name of the user to whom job related mail was sent

projectName (%s)

LSF project name

exitStatus (%d)

UNIX exit status of the job

maxNumProcessors (%d)

Maximum number of processors specified for the job

loginShell (%s)

Login shell used for the job

timeEvent (%s)

Time event string for the job - IBM Spectrum LSF Process Manager only

idx (%d)

Job array index

maxRMem (%d)

Maximum resident memory usage in KB of all processes in the job

maxRSwap (%d)

Maximum virtual memory usage in KB of all processes in the job

inFileSpool (%s)

Spool input file (up to 4094 characters for UNIX or 512 characters for Windows)

commandSpool (%s)

Spool command file (up to 4094 characters for UNIX or 512 characters for Windows)

rsvId %s

Advance reservation ID for a user group name less than 120 characters long; for example, "user2#0"

If the advance reservation user group name is longer than 120 characters, the rsvId field output appears last.

sla (%s)

SLA service class name under which the job runs

exceptMask (%d)

Job exception handling

Values:

- J_EXCEPT_OVERRUN 0x02
- J_EXCEPT_UNDERUN 0x04
- J_EXCEPT_IDLE 0x80

additionalInfo (%s)

Placement information of HPC jobs

exitInfo (%d)

Job termination reason, mapped to corresponding termination keyword displayed by **bacct**.

warningAction (%s)

Job warning action

warningTimePeriod (%d)

Job warning time period in seconds

chargedSAAP (%s)

SAAP charged to a job

licenseProject (%s)

IBM Spectrum LSF License Scheduler project name

app (%s)

Application profile name

postExecCmd (%s)

Post-execution command to run on the execution host after the job finishes

runtimeEstimation (%d)

Estimated run time for the job, calculated as the CPU factor of the submission host multiplied by the runtime estimate (in seconds).

jobGroupName (%s)

Job group name

requeueValues (%s)

Requeue exit value

options2 (%d)

Bit flags for job processing

resizeNotifyCmd (%s)

Resize notification command to be invoked on the first execution host upon a resize request.

lastResizeTime (%d)

Last resize time. The latest wall clock time when a job allocation is changed.

rsvId %s

Advance reservation ID for a user group name more than 120 characters long.

If the advance reservation user group name is longer than 120 characters, the rsvId field output appears last.

jobDescription (%s)

Job description (up to 4094 characters).

submitEXT

Submission extension field, reserved for internal use.

Num (%d)

Number of elements (key-value pairs) in the structure.

key (%s)

Reserved for internal use.

value (%s)

Reserved for internal use.

options3 (%d)

Bit flags for job processing

bsub -W(%d)

Job submission runtime limit

numHostRusage(%d)

The number of host-based resource usage entries (*hostRusage*) that follow. Enabled by default.

hostRusage

The following fields contain host-based resource usage information for the job. To disable reporting of *hostRusage* set **LSF_HPC_EXTENSIONS=NO_HOST_RUSAGE** in *lsf.conf*.

hostname (%s)

Name of the host.

mem(%d)

Total resident memory usage of all processes in the job running on this host.

swap(%d)

The total virtual memory usage of all processes in the job running on this host.

utime(%d)

User time used on this host.

stime(%d)

System time used on this host.

hHostExtendInfo(%d)

Number of following key-value pairs containing extended host information (PGIDs and PIDs). Set to 0 in *lsb.events*, *lsb.acct*, and *lsb.stream* files.

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

effectiveResReq (%s)

The runtime resource requirements used for the job.

network (%s)

Network requirements for IBM Parallel Environment (PE) jobs.

runTime (%d)

Time in seconds that the job has been in the run state. runTime includes the totalProvisionTime.

cpu_frequency(%d)

CPU frequency at which the job ran.

options4 (%d)

Bit flags for job processing

numAllocSlots(%d)

Number of allocated slots.

allocSlots(%s)

List of execution host names where the slots are allocated.

ineligiblePendTime(%d)

Time in seconds that the job has been in the ineligible pending state.

indexRangeCnt (%d)

The number of element ranges indicating successful signals

indexRangeStart1 (%d)

The start of the first index range.

indexRangeEnd1 (%d)

The end of the first index range.

indexRangeStep1 (%d)

The step of the first index range.

indexRangeStartN (%d)

The start of the last index range.

indexRangeEndN (%d)

The end of the last index range.

indexRangeStepN (%d)

The step of the last index range.

userGroup (%s)

The user group.

EVENT_ADRSV_FINISH

An advance reservation has expired. The fields in order of occurrence are:

Event type (%s)

Which is EVENT_ADRSV_FINISH

Version Number (%s)

Version number of the log file format

Event Logging Time (%d)

Time the event was logged (in seconds since the epoch); for example, "1038942015"

Reservation Creation Time (%d)

Time the advance reservation was created (in seconds since the epoch); for example, 1038938898

Reservation Type (%d)

Type of advance reservation request:

- User reservation (RSV_OPTION_USER, defined as 0x001)
- User group reservation (RSV_OPTION_GROUP, defined as 0x002)
- System reservation (RSV_OPTION_SYSTEM, defined as 0x004)
- Recurring reservation (RSV_OPTION_RECUR, defined as 0x008)

For example, 9 is a recurring reservation created for a user.

Creator ID (%d)

UNIX user ID of the reservation creator; for example, 30408

Reservation ID (rsvId %s)

For example, user2#0

User Name (%s)

User name of the reservation user; for example, user2

Time Window (%s)

Time window of the reservation:

- One-time reservation in seconds since the epoch; for example, 1033761000-1033761600
- Recurring reservation; for example, 17:50-18:00

Creator Name (%s)

User name of the reservation creator; for example, user1

Duration (%d)

Duration of the reservation, in hours, minutes, seconds; for example, 600 is 6 hours, 0 minutes, 0 seconds

Number of Resources (%d)

Number of reserved resource pairs in the resource list; for example 2 indicates 2 resource pairs (hostA 1 hostB 1)

Host Name (%s)

Reservation host name; for example, hostA

Number of CPUs (%d)

Number of reserved CPUs; for example 1

JOB_RESIZE

When there is an allocation change, LSF logs the event after **mbatchd** receives a JOB_RESIZE_NOTIFY_DONE event. From lastResizeTime and eventTime, you can calculate the duration of previous job allocation. The fields in order of occurrence are:

Version number (%s)

The version number.

Event Time (%d)

Time the event was logged (in seconds since the epoch).

jobId (%d)

ID for the job.

tdx (%d)

Job array index.

startTime (%d)

The start time of the running job.

userId (%d)

UNIX user ID of the user invoking the command

userName (%s)

User name of the submitter

resizeType (%d)

Resize event type, 0, grow, 1 shrink.

lastResizeTime(%d)

The wall clock time when job allocation is changed previously. The first lastResizeTime is the job start time.

numExecHosts (%d)

The number of execution hosts before allocation is changed. Support LSF_HPC_EXTENSIONS="SHORT_EVENTFILE".

execHosts (%s)

Execution host list before allocation is changed. Support LSF_HPC_EXTENSIONS="SHORT_EVENTFILE".

numResizeHosts (%d)

Number of processors used for execution during resize. If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is the number of hosts listed in short format.

resizeHosts (%s)

List of execution host names during resize. If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is logged in a shortened format.

numAllocSlots(%d)

Number of allocated slots.

allocSlots(%s)

List of execution host names where the slots are allocated.

numResizeSlots (%d)

Number of allocated slots for executing resize.

resizeSlots (%s)

List of execution host names where slots are allocated for resizing.

lsb.applications

The lsb.applications file defines application profiles. Use application profiles to define common parameters for the same type of jobs, including the execution requirements of the applications, the resources they require, and how they should be run and managed.

This file is optional. Use the **DEFAULT_APPLICATION** parameter in the lsb.params file to specify a default application profile for all jobs. LSF does not automatically assign a default application profile.

This file is installed by default in the LSB_CONFDIR/*cluster_name*/configdir directory.

Changing lsb.applications configuration

After you change the lsb.applications file, run the **badadmin reconfig** command to reconfigure the **mbatchd** daemon. Configuration changes apply to pending jobs only. Running jobs are not affected.

lsb.applications structure

Each application profile definition begins with the line **Begin Application** and ends with the line **End Application**. The application name must be specified. All other parameters are optional.

Example

```
Begin Application
NAME           = catia
DESCRIPTION    = CATIA V5
CPULIMIT       = 24:0/hostA      # 24 hours of host hostA
FILELIMIT      = 20000
DATALIMIT      = 20000          # jobs data segment limit
CORELIMIT      = 20000
TASKLIMIT      = 5              # job processor limit
REQUEUE_EXIT_VALUES = 55 34 78
End Application
```

See the `lsb.applications` template file for other application profile examples.

#INCLUDE

Syntax

```
#INCLUDE "path-to-file"
```

Description

Inserts a configuration setting from another file to the current location. Use this directive to dedicate control of a portion of the configuration to other users or user groups by providing write access for the included file to specific users or user groups, and to ensure consistency of configuration file settings in different clusters (if you are using the LSF multicluster capability).

See more information on shared configuration file content in *Administering IBM Spectrum LSF*

#INCLUDE can be inserted anywhere in the local configuration file.

Default

Not defined.

ABS_RUNLIMIT

Syntax

```
ABS_RUNLIMIT=y | Y
```

Description

If set, absolute (wall-clock) run time is used instead of normalized run time for all jobs submitted with the following values:

- Run time limit or run time estimate specified by the `-W` or `-We` option of **bsub**
- RUNLIMIT queue-level parameter in `lsb.queues`
- RUNLIMIT application-level parameter in `lsb.applications`
- ESTIMATED_RUNTIME parameter in `lsb.applications`

The runtime estimates and limits are not normalized by the host CPU factor.

Default

Not defined. Run limit and runtime estimate are normalized.

BIND_JOB

BIND_JOB specifies the processor binding policy for sequential and parallel job processes that run on a single host. On Linux execution hosts that support this feature, job processes are hard bound to selected processors.

Syntax

```
BIND_JOB=NONE | BALANCE | PACK | ANY | USER | USER_CPU_LIST
```

Description

Note: **BIND_JOB** is deprecated in LSF Standard Edition and LSF Advanced Edition. You should enable LSF CPU and memory affinity scheduling in with the `AFFINITY` parameter in `lsb.hosts`. If both `BIND_JOB` and affinity scheduling are enabled, affinity scheduling takes effect, and `LSF_BIND_JOB` is disabled.

BIND_JOB and **LSF_BIND_JOB** are the only affinity options available in LSF Express Edition.

If processor binding feature is not configured with the **BIND_JOB** parameter in an application profile in `lsb.applications`, the **LSF_BIND_JOB** configuration setting `lsf.conf` takes effect. The application profile configuration for processor binding overrides the `lsf.conf` configuration.

For backwards compatibility:

- `BIND_JOB=Y` is interpreted as `BIND_JOB=BALANCE`
- `BIND_JOB=N` is interpreted as `BIND_JOB=NONE`

Supported platforms

Linux with kernel version 2.6 or higher

Default

Not defined. Processor binding is disabled.

CHKPNT_DIR

Syntax

`CHKPNT_DIR=chkpnt_dir`

Description

Specifies the checkpoint directory for automatic checkpointing for the application. To enable automatic checkpoint for the application profile, administrators must specify a checkpoint directory in the configuration of the application profile.

If `CHKPNT_PERIOD`, `CHKPNT_INITPERIOD` or `CHKPNT_METHOD` was set in an application profile but `CHKPNT_DIR` was not set, a warning message is issued and those settings are ignored.

The checkpoint directory is the directory where the checkpoint files are created. Specify an absolute path or a path relative to the current working directory for the job. Do not use environment variables in the directory path.

If checkpoint-related configuration is specified in the queue, application profile, and at job level:

- Application-level and job-level parameters are merged. If the same parameter is defined at both job-level and in the application profile, the job-level value overrides the application profile value.
- The merged result of job-level and application profile settings override queue-level configuration.

To enable checkpointing of jobs with the LSF multicluster capability, define a checkpoint directory in an application profile (`CHKPNT_DIR`, `CHKPNT_PERIOD`, `CHKPNT_INITPERIOD`, `CHKPNT_METHOD` in `lsb.applications`) of both submission cluster and execution cluster. LSF uses the directory specified in the execution cluster.

Checkpointing is not supported if a job runs on a leased host.

The file path of the checkpoint directory can contain up to 4000 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

Default

Not defined

CHKPNT_INITPERIOD

Syntax

`CHKPNT_INITPERIOD=init_chkpnt_period`

Description

Specifies the initial checkpoint period in minutes. `CHKPNT_DIR` must be set in the application profile for this parameter to take effect. The periodic checkpoint specified by `CHKPNT_PERIOD` does not happen until the initial period has elapsed.

Specify a positive integer.

Job-level command line values override the application profile configuration.

If administrators specify an initial checkpoint period and do not specify a checkpoint period (`CHKPNT_PERIOD`), the job will only checkpoint once.

If the initial checkpoint period of a job is specified, and you run **`bchkpnt`** to checkpoint the job at a time before the initial checkpoint period, the initial checkpoint period is not changed by **`bchkpnt`**. The first automatic checkpoint still happens after the specified number of minutes.

Default

Not defined

CHKPNT_PERIOD**Syntax**

`CHKPNT_PERIOD=chkpnt_period`

Description

Specifies the checkpoint period for the application in minutes. `CHKPNT_DIR` must be set in the application profile for this parameter to take effect. The running job is checkpointed automatically every checkpoint period.

Specify a positive integer.

Job-level command line values override the application profile and queue level configurations. Application profile level configuration overrides the queue level configuration.

Default

Not defined

CHKPNT_METHOD**Syntax**

`CHKPNT_METHOD=chkpnt_method`

Description

Specifies the checkpoint method. `CHKPNT_DIR` must be set in the application profile for this parameter to take effect. Job-level command line values override the application profile configuration.

Default

Not defined

CHUNK_JOB_SIZE**Syntax**

`CHUNK_JOB_SIZE=integer`

Description

Chunk jobs only. Allows jobs submitted to the same application profile to be chunked together and specifies the maximum number of jobs allowed to be dispatched together in a chunk. Specify a positive integer greater than or equal to 1.

All of the jobs in the chunk are scheduled and dispatched as a unit, rather than individually.

Specify `CHUNK_JOB_SIZE=1` to disable job chunking for the application. This value overrides chunk job dispatch configured in the queue.

Use the `CHUNK_JOB_SIZE` parameter to configure application profiles that chunk small, short-running jobs. The ideal candidates for job chunking are jobs that have the same host and resource requirements and typically take 1 to 2 minutes to run.

Job chunking can have the following advantages:

- Reduces communication between sbatchd and mbatchd and reduces scheduling overhead in mbschd.
- Increases job throughput in mbatchd and CPU utilization on the execution hosts.

However, throughput can deteriorate if the chunk job size is too big. Performance may decrease on profiles with `CHUNK_JOB_SIZE` greater than 30. You should evaluate the chunk job size on your own systems for best performance.

With the job forwarding model for the LSF multicluster capability, this parameter does not affect jobs that are forwarded to a remote cluster.

LSF ignores this parameter does not chunk jobs under the following conditions:

- CPU limit greater than 30 minutes (**CPULIMIT** parameter in `lsb.queues` or `lsb.applications`)
- Run limit greater than 30 minutes (**RUNLIMIT** parameter in `lsb.queues` or `lsb.applications`)
- Runtime estimate greater than 30 minutes (**ESTIMATED_RUNTIME** parameter in `lsb.applications`)

If **CHUNK_JOB_DURATION** is set in `lsb.params`, chunk jobs are accepted regardless of the value of **CPULIMIT**, **RUNLIMIT** or **ESTIMATED_RUNTIME**.

Default

Not defined

CONTAINER**Syntax**

`CONTAINER=docker[image(image_name) options(docker_run_options)]`

`CONTAINER=nvidia-docker[image(image_name) options(docker_run_options)]`

`CONTAINER=shifter[image(image_name) options(container_options)]`

`CONTAINER=singularity[image(image_name) options(container_options)]`

Description

Enables LSF to use a Docker, NVIDIA Docker, Shifter, or Singularity container for jobs that are submitted to this application profile. This parameter uses the following keywords:

docker | nvidia-docker | shifter | singularity

Required. Use one of these keywords to specify the type of container to use for jobs that are submitted to this application profile.

image

Required. This keyword specifies the image name that is used in running jobs.

For Docker and NVIDIA Docker jobs, use the **\$LSB_CONTAINER_IMAGE** environment variable to allow users to specify the image name for Docker and NVIDIA Docker container jobs at job submission time.

At job submission time, users can specify a specific image name that is in the specified repository server by specifying a value for the **\$LSB_CONTAINER_IMAGE** environment variable.

options

Optional. This keyword specifies the job run options for the container.

To enable a pre-execution script to run, specify an at sign (@) and a full file path to the script, which the execution host must be able to access. Before the container job runs, LSF runs this script with LSF administrator privileges. While the script is running, the jobs' environment variables are passed to the script. When the script finishes running, the output is used as container startup options. The script must provide this output on one line. The method of processing the container job depends on the result of the pre-execution script:

- If the pre-execution script failed, the container job exits with the same exit code from the script. In addition, an external status message is sent to inform the user that the job exited because of script execution failure.
- If the execution of the script is successful but the output contains more than 512 options, LSF only keeps the first 512 options, and the remaining options are ignored.
- If the execution of the script is successful and the output is valid, the output is part of the container job running options. The position of the output from the script in the options is exactly where the user configured the script in the options field.

For Docker and NVIDIA Docker containers, this keyword specifies the Docker job run options for the **docker run** command, which are passed to the job container.

Note:

- Before you specify the Docker job run options, make sure that these options work with the **docker run** command in the command line.
- The `--cgroup-parent`, `--user` (or `-u`), and `--name` options are reserved for LSF internal use. Do not use these options in the options keyword configuration, otherwise the job fails.
If you specified a pre-execution script and the output of this script contains `--cgroup-parent`, `--user`, `-u`, or `--name`, the container job also fails.
- The `-w` and `--ulimit` options are automatically set for LSF. Do not use these options in the options keyword configuration because these specifications override the LSF settings.
- The `-v` option is automatically used by LSF to mount the working directories that LSF requires, such as the current working directory, job spool directory, destination file for the **bsub -f** command, `tmp` directory, the `LSF_TOP`, and the checkpoint directory on demand.
- You can configure the `--rm` option in the options keyword configuration to automatically remove containers after the job is done.
- Limitation: If you use the `-d` option, LSF incorrectly gets the status of the Docker jobs as `DONE`.

For Shifter containers, this keyword specifies the Shifter job run options for the **shifter** command, which are passed to the job container.

Note:

- Run **shifter --help** in the command line to view the options that the **shifter** command supports.
- Before you specify the Shifter job run options, make sure that these options work with the **shifter** command in the command line.
- The **\$LD_LIBRARY_PATH** directory is cleaned according to the `setuid` bit that Shifter uses to work. Therefore, for programs that depend on **\$LD_LIBRARY_PATH** to work (such as **openmpi**), ensure that the `setuid` bit can be properly set inside the container by adding **LD_LIBRARY_PATH** to the **siteEnvAppend** section of the `udiRoot.conf` file.

For Singularity containers, this keyword specifies the Singularity job run options for the **singularity exec** command, which are passed to the job container.

Note:

- Run **singularity exec --help** in the command line to view the options that the **singularity** command supports.
- Before you specify the Singularity job run options, make sure that these options work with the **singularity exec** command in the command line.
- The **\$LD_LIBRARY_PATH** directory is cleaned according to the setuid bit that Singularity uses to work. Therefore, for programs that depend on **\$LD_LIBRARY_PATH** to work (such as **openmpi**), ensure that the setuid bit can be properly set inside the container by adding **LD_LIBRARY_PATH** to the **ld.so.conf** file and run the **ldconfig** command.

Examples

To specify an Ubuntu image for use with container jobs without specifying any optional keywords,

```
Begin Application
NAME = dockerapp
CONTAINER = docker[image(repository.example.com:5000/file/path/ubuntu:latest)]
DESCRIPTION = Docker User Service
End Application
```

```
Begin Application
NAME = ndockerapp
CONTAINER = nvidia-docker[image(repository.example.com:5000/file/path/ubuntu:latest)]
DESCRIPTION = NVIDIA Docker User Service
End Application
```

```
Begin Application
NAME = shifterapp
CONTAINER = shifter[image(ubuntu:latest)]
DESCRIPTION = Shifter User Service
End Application
```

```
Begin Application
NAME = singapp
CONTAINER = singularity[image(/file/path/ubuntu.img)]
DESCRIPTION = Singularity User Service
End Application
```

To specify a pre-execution script in the **/share/usr/** directory, which generates the container startup options,

```
Begin Application
NAME = dockerappoptions
CONTAINER = docker[image(repository.example.com:5000/file/path/ubuntu:latest) options(@/share/usr/docker-options.sh)]
DESCRIPTION = Docker User Service with pre-execution script for options
End Application
```

```
Begin Application
NAME = ndockerappoptions
CONTAINER = nvidia-docker[image(repository.example.com:5000/file/path/ubuntu:latest) options(@/share/usr/docker-options.sh)]
DESCRIPTION = NVIDIA Docker User Service with pre-execution script for options
End Application
```

```
Begin Application
NAME = shifterappoptions
CONTAINER = shifter[image(ubuntu:latest) options(@/share/usr/shifter-options.sh)]
DESCRIPTION = Shifter User Service
End Application
```

```
Begin Application
NAME = singappoptions
CONTAINER = singularity[image(/file/path/ubuntu.img) options(@/share/usr/sing-options.sh)]
DESCRIPTION = Singularity User Service
End Application
```

- For sequential jobs, specify the following **CONTAINER** parameter value for LSF to automatically remove containers after the job is done:

```
CONTAINER=docker[image(image-name) options(--rm)]
```

- For parallel jobs, the network and IPC must work across containers to make **blaunch** work. The execution user ID and user name mapping file must be mounted to the container for **blaunch** authentication.

Therefore, specify the following **CONTAINER** parameter value for LSF to configure the container IPC and network parameters so that **blaunch** can work across multiple containers, to configure the container password file for **blaunch** authentication, and automatically remove containers after the job is done:

```
CONTAINER=docker[image(image-name) options(--rm --network=host --ipc=host -v /path/to/my/passwd:/etc/passwd)]
```

The passwd file must be in the standard format for UNIX and Linux password files, such as the following format:

```
user1:x:10001:10001:::
user2:x:10002:10002:::
```

- To allow users to specify image names for Docker and NVIDIA Docker container jobs at job submission time, use the **\$LSB_CONTAINER_IMAGE** environment variable as the image name when specifying the **image** keyword.

For example, when defining the **CONTAINER** parameter for the udocker application profile, add the **\$LSB_CONTAINER_IMAGE** environment variable to the image specification:

```
Begin Application
NAME = udockerGPU
CONTAINER = docker[image(repository.example.com:5000/$LSB_CONTAINER_IMAGE) \
              options(--rm --net=host --ipc=host -v /gpfs/u/fred:/data )]
DESCRIPTION = Docker User Service
End Application
```

Specify a container image name (such as ubuntu) at the job submission time by setting the **\$LSB_CONTAINER_IMAGE** environment using one of the following methods:

- Specify the **\$LSB_CONTAINER_IMAGE** environment variable according to your shell environment:

- In **csh** or **tcsh**:

```
setenv LSB_CONTAINER_IMAGE ubuntu
```

- In **sh**, **ksh**, or **bash**:

```
export LSB_CONTAINER_IMAGE=ubuntu
```

- Use the **bsub -env** option:

```
bsub -env LSB_CONTAINER_IMAGE=ubuntu -app udocker a.out -in in.dat -out out.dat
```

- Use an **esub** script to set the **LSB_CONTAINER_IMAGE** environment variable, then call the **esub** with the **bsub** command.

For example, create an **esub.docker** script in the **\$LSF_SERVERDIR** directory with the following contents:

```
#!/bin/sh
exec 1>&2
echo "LSB_CONTAINER_IMAGE=\"$1\" " >> $LSB_SUB_MODIFY_ENVFILE
```

Submit a job to call the **esub.docker** script by running the following command:

```
bsub -a "docker(ubuntu)" -app udocker a.out -in in.dat -out out.dat
```

Default

Undefined

CORELIMIT

Syntax

CORELIMIT=*integer*

Description

The per-process (soft) core file size limit for all of the processes belonging to a job from this application profile (see **getrlimit(2)**). Application-level limits override any default limit specified in the queue, but must be less than the hard limit of the submission queue. Job-level core limit (**bsub -C**) overrides queue-level and application-level limits.

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

Default

Unlimited

CPU_FREQUENCY

Syntax

CPU_FREQUENCY=*[float_number][unit]*

Description

Specifies the CPU frequency for an application profile. All jobs submit to the application profile require the specified CPU frequency. Value is a positive float number with units (GHz, MHz, or KHz). If no units are set, the default is GHz.

This value can also be set using the command **bsub -freq**.

The submission value will overwrite the application profile value, and the application profile value will overwrite the queue value.

Default

Not defined (Nominal CPU frequency is used)

CPULIMIT

Syntax

CPULIMIT=*[[:hour:]minute[/host_name | /host_model]*

Description

Limits the total CPU time the job can use. This parameter is useful for preventing runaway jobs or jobs that use up too many resources.

When the total CPU time for the whole job has reached the limit, a SIGXCPU signal is sent to all processes belonging to the job. If the job has no signal handler for SIGXCPU, the job is killed immediately. If the SIGXCPU signal is handled, blocked, or ignored by the application, then after the grace period expires, LSF sends SIGINT, SIGTERM, and SIGKILL to the job to kill it.

If a job dynamically spawns processes, the CPU time used by these processes is accumulated over the life of the job.

Processes that exist for fewer than 30 seconds may be ignored.

By default, jobs submitted to the application profile without a job-level CPU limit (**bsub -c**) are killed when the CPU limit is reached. Application-level limits override any default limit specified in the queue.

The number of minutes may be greater than 59. For example, three and a half hours can be specified either as 3:30 or 210.

If no host or host model is given with the CPU time, LSF uses the default CPU time normalization host defined at the queue level (DEFAULT_HOST_SPEC in `lsb.queues`) if it has been configured, otherwise uses the default CPU time normalization host defined at the cluster level (DEFAULT_HOST_SPEC in `lsb.params`) if it has been configured, otherwise uses the host with the largest CPU factor (the fastest host in the cluster).

On Windows, a job that runs under a CPU time limit may exceed that limit by up to SBD_SLEEP_TIME. This is because sbatchd periodically checks if the limit has been exceeded.

On UNIX systems, the CPU limit can be enforced by the operating system at the process level.

You can define whether the CPU limit is a per-process limit enforced by the OS or a per-job limit enforced by LSF with LSB_JOB_CPULIMIT in `lsf.conf`.

Default

Unlimited

DATALIMIT

Syntax

DATALIMIT=*integer*

Description

The per-process (soft) data segment size limit (in KB) for all of the processes belonging to a job running in the application profile (see **getrlimit(2)**).

By default, jobs submitted to the application profile without a job-level data limit (**bsub -D**) are killed when the data limit is reached. Application-level limits override any default limit specified in the queue, but must be less than the hard limit of the submission queue.

Default

Unlimited

DESCRIPTION

Syntax

DESCRIPTION=*text*

Description

Description of the application profile. The description is displayed by **bapp -l**.

The description should clearly describe the service features of the application profile to help users select the proper profile for each job.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\). The maximum length for the text is 512 characters.

DJOB_COMMFAIL_ACTION

Syntax

DJOB_COMMFAIL_ACTION="KILL_TASKS | IGNORE_COMMFAIL "

Description

Defines the action LSF should take if it detects a communication failure with one or more remote parallel or distributed tasks. If defined with "KILL_TASKS", LSF tries to kill all the current tasks of a parallel or distributed job associated with the communication failure. If defined with "IGNORE_COMMFAIL", failures will be ignored and the job continues. If not defined, LSF terminates all tasks and shuts down the entire job.

This parameter only applies to the **blaunch** distributed application framework.

When defined in an application profile, the LSB_DJOB_COMMFAIL_ACTION variable is set when running **bsub -app** for the specified application.

Default

Not defined. Terminate all tasks, and shut down the entire job.

DJOB_DISABLED

Syntax

DJOB_DISABLED=Y | N

Description

Disables the **blaunch** distributed application framework.

Default

Not defined. Distributed application framework is enabled.

DJOB_ENV_SCRIPT

Syntax

DJOB_ENV_SCRIPT=*script_name*

Description

Defines the name of a user-defined script for setting and cleaning up the parallel or distributed job environment.

The specified script must support a setup argument and a cleanup argument. The script is executed by LSF with the setup argument before launching a parallel or distributed job, and with argument cleanup after the job is finished.

The script runs as the user, and is part of the job.

If a full path is specified, LSF uses the path name for the execution. Otherwise, LSF looks for the executable from \$LSF_BINDIR.

This parameter only applies to the **blaunch** distributed application framework.

When defined in an application profile, the LSB_DJOB_ENV_SCRIPT variable is set when running **bsub -app** for the specified application.

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

If DJOB_ENV_SCRIPT=openmpi_rankfile.sh is set in lsb.applications, LSF creates a host rank file and sets the environment variable LSB_RANK_HOSTFILE.

Default

Not defined.

DJOB_HB_INTERVAL

Syntax

DJOB_HB_INTERVAL=*seconds*

Description

Value in seconds used to calculate the heartbeat interval between the task RES and job RES of a parallel or distributed job.

This parameter only applies to the **blaunch** distributed application framework.

When DJOB_HB_INTERVAL is specified, the interval is scaled according to the number of tasks in the job:

$$\max(\text{DJOB_HB_INTERVAL}, 10) + \text{host_factor}$$

where

$$\text{host_factor} = 0.01 * \text{number of hosts allocated for the job}$$

Default

Not defined. Interval is the default value of **LSB_DJOB_HB_INTERVAL**.

DJOB_RESIZE_GRACE_PERIOD

Syntax

DJOB_RESIZE_GRACE_PERIOD = *seconds*

Description

When a resizable job releases resources, the LSF distributed parallel job framework terminates running tasks if a host has been completely removed. A **DJOB_RESIZE_GRACE_PERIOD** defines a grace period in seconds for the application to clean up tasks itself before LSF forcibly terminates them.

Default

No grace period.

DJOB_RU_INTERVAL

Syntax

DJOB_RU_INTERVAL=*seconds*

Description

Value in seconds used to calculate the resource usage update interval for the tasks of a parallel or distributed job.

This parameter only applies to the **blaunch** distributed application framework.

When DJOB_RU_INTERVAL is specified, the interval is scaled according to the number of tasks in the job:

$$\max(\text{DJOB_RU_INTERVAL}, 10) + \text{host_factor}$$

where

$$\text{host_factor} = 0.01 * \text{number of hosts allocated for the job}$$

Default

Not defined. Interval is the default value of **LSB_DJOB_RU_INTERVAL**.

DJOB_TASK_BIND

Syntax

DJOB_TASK_BIND=Y | y | N | n

Description

For CPU and memory affinity scheduling jobs launched with the **blaunch** distributed application framework.

To enable LSF to bind each task to the proper CPUs or NUMA nodes you must use **blaunch** to start tasks. You must set DJOB_TASK_BIND=Y in `lsb.applications` or `LSB_DJOB_TASK_BIND=Y` in the submission environment before submitting the job. When set, only the CPU and memory bindings allocated to the task itself will be set in each task's environment.

If DJOB_TASK_BIND=N or `LSB_DJOB_TASK_BIND=N`, or they are not set, each task will have the same CPU or NUMA node binding on one host.

If you do not use **blaunch** to start tasks, and use another MPI mechanism such as IBM Spectrum LSF MPI or IBM Parallel Environment, you should not set **DJOB_TASK_BIND** or set it to N.

Default

N

DOCKER_IMAGE_AFFINITY

Syntax

DOCKER_IMAGE_AFFINITY=Y | y | N | n

Description

When scheduling Docker-based containerized jobs, setting this parameter to y or Y enables LSF to give preference for execution hosts that already have the requested Docker image. This reduces network bandwidth and the job start time because the execution host does not have to pull the Docker image from the repository and the job can immediately start on the execution host.

When this feature is enabled, LSF considers Docker image location information when scheduling Docker jobs. Docker image affinity interacts with host preference and **order[]** string requests in the following manner:

- If host preference is specified, the host preference is honored first. Among hosts with the same preference level, hosts with the requested Docker image are given higher priority.
- If the **order[]** string is specified, the hosts with the requested Docker image have a higher priority first. Among hosts that all have the requested Docker image, the **order[]** string is then honored.

The **CONTAINER** parameter must be defined for this parameter to work with this application profile.

Default

Not defined.

ELIGIBLE_PEND_TIME_LIMIT**Syntax**

`ELIGIBLE_PEND_TIME_LIMIT=[hour:]minute`

Description

Specifies the eligible pending time limit for a job.

LSF sends the application-level eligible pending time limit configuration to IBM Spectrum LSF RTM, which handles the alarm and triggered actions such as user notification (for example, notifying the user that submitted the job and the LSF administrator) and job control actions (for example, killing the job). IBM Spectrum LSF RTM compares the job's eligible pending time to the eligible pending time limit, and if the job is in an eligible pending state for longer than this specified time limit, IBM Spectrum LSF RTM triggers the alarm and actions. This parameter works without IBM Spectrum LSF RTM, but LSF does not take any other alarm actions.

In the job forwarding model for the LSF multicluster capability, the job's eligible pending time limit is ignored in the execution cluster, while the submission cluster merges the job's queue-, application-, and job-level eligible pending time limit according to local settings.

The eligible pending time limit is in the form of `[hour:]minute`. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The job-level eligible pending time limit (**bsub -ept1**) overrides the application-level limit specified here, and the application-level limit overrides the queue-level limit (**ELIGIBLE_PEND_TIME_LIMIT** in `lsb.queue`s).

Default

Not defined.

ENV_VARS**Syntax**

`ENV_VARS="name='value'[,name1='value1'][,name2='value2',...]"`

Description

ENV_VARS defines application-specific environment variables that will be used by jobs for the application. Use this parameter to define name/value pairs as environment variables. These environment variables are also used in the pre/post-execution environment.

You can include spaces within the single quotation marks when defining a value. Commas and double quotation marks are reserved by LSF and cannot be used as part of the environment variable name or value. If the same environment variable is named multiple times in **ENV_VARS** and given different values, the last value in the list will be the one which takes effect. LSF does not allow environment variables to contain other environment variables to be expanded on the execution side. Do not redefine LSF environment variables in **ENV_VARS**.

To define a NULL environment variable, use single quotes with nothing inside. For example:

`ENV_VARS="TEST_CAR= ' ' "`

Any variable set in the user's environment will overwrite the value in **ENV_VARS**. The application profile value will overwrite the execution host environment value.

After changing the value of this parameter, run **badadmin reconfig** to have the changes take effect. The changes apply to pending jobs only. Running jobs are not affected.

Default

Not defined.

ESTIMATED_RUNTIME

Syntax

```
ESTIMATED_RUNTIME=[hour:]minute[/host_name | /host_model]
```

Description

This parameter specifies an estimated run time for jobs associated with an application. LSF uses the ESTIMATED_RUNTIME value for scheduling purposes only, and does not kill jobs that exceed this value unless the jobs also exceed a defined RUNLIMIT. The format of runtime estimate is same as the RUNLIMIT parameter.

The job-level runtime estimate specified by **bsub -We** overrides the ESTIMATED_RUNTIME setting in an application profile. The ESTIMATED_RUNTIME setting in an application profile overrides the ESTIMATED_RUNTIME setting in the queue and cluster.

The following LSF features use the ESTIMATED_RUNTIME value to schedule jobs:

- Job chunking
- Advance reservation
- SLA
- Slot reservation
- Backfill
- Allocation planner

Default

Not defined

EXEC_DRIVER

Syntax

```
EXEC_DRIVER=context[user(user_name)] starter[/file_path_serverdir/docker-starter.py]  
controller[/file_path/to/serverdir/docker-control.py] monitor[/file_path/to/serverdir/  
docker-monitor.py]
```

Replace *file_path/to/serverdir* with the actual file path of the *LSF_SERVERDIR* directory.

Description

Specifies the execution driver framework for Docker container jobs in this application profile. This parameter uses the following keyword:

user

Optional. This keyword specifies the user account for starting scripts. The configured value is a user name instead of a user ID. For Docker jobs, this user must be a member of the Docker user group. By default this is the LSF primary administrator.

Note: This cannot be the root user.

LSF includes three execution driver scripts that are used to start a job (`docker-starter.py`), monitor the resource of a job (`docker-monitor.py`), and send a signal to a job (`docker-control.py`). These scripts are located in the `LSF_SERVERDIR` directory. Change the owner of the script files to the context user and change the file permissions to 700 or 500 before using them in the **EXEC_DRIVER** parameter.

Interaction with the **CONTAINER** parameter for Docker jobs

For Docker jobs, the **EXEC_DRIVER** parameter interacts with the following keywords in the **CONTAINER** parameter:

- **image**, which specifies the image name (`$LSB_CONTAINER_IMAGE` environment variable) is supported when specifying the script names.
- **options** with runtime options and the option script is supported.

Example

```
Begin Application
NAME = dockerapp
CONTAINER = docker[image(repository.example.com:5000/file/path/ubuntu:latest)
               options(--rm --network=host --ipc=host -v /path/to/my/passwd:/etc/passwd)]
EXEC_DRIVER = context[user(user-name)] starter[/path/to/driver/docker-starter.py]
               controller[/path/to/driver/docker-control.py]
               monitor[/path/to/driver/docker-monitor.py]
DESCRIPTION = Docker User Service
End Application
```

Default

Undefined

FILELIMIT

Syntax

`FILELIMIT=integer`

Description

The per-process (soft) file size limit (in KB) for all of the processes belonging to a job running in the application profile (see `getrlimit(2)`). Application-level limits override any default limit specified in the queue, but must be less than the hard limit of the submission queue.

Default

Unlimited

GPU_REQ

Specify GPU requirements together in one statement.

Syntax

```
GPU_REQ = "[num=num_gpus[/task | host]] [:mode=shared | exclusive_process]
[:mps=yes[, shared]] no | per_socket[, shared] | per_gpu[, shared]] [:j_exclusive=yes | no]
[:gmodel=model_name[#mem_size]] [:gtile=tile_num|'!'] [:gmem=mem_value] [:nvlink=yes]
[:aff=yes | no]"
```

Description

The `LSB_GPU_NEW_SYNTAX=extend` or `LSB_GPU_NEW_SYNTAX=Y` parameter must be set in the `lsf.conf` file to specify the `j_exclusive`, `mode`, `mps`, and `num` GPU requirements in the **GPU_REQ** parameter.

The **LSB_GPU_NEW_SYNTAX=extend** must be set in the `lsf.conf` file to specify the `gmem`, `gmodel`, `gtile`, and `nvlink` GPU requirements.

When the **LSB_GPU_NEW_SYNTAX=extend** parameter is set, the resource requirements of your job cannot use the legacy GPU resources (`ngpus_shared`, `ngpus_excl_t`, `ngpus_excl_p`) as job resource requirements.

The **GPU_REQ** parameter takes the following arguments:

num=num_gpus[/task | host]

The number of physical GPUs required by the job. By default, the number is per host. You can also specify that the number is per task by specifying `/task` after the number.

If you specified that the number is per task, the configuration of the `ngpus_physical` resource in the `lsb.resources` file is set to `PER_TASK`, or the **RESOURCE_RESERVE_PER_TASK=Y** parameter is set in the `lsb.params` file, this number is the requested count per task.

mode=shared | exclusive_process

The GPU mode when the job is running, either `shared` or `exclusive_process`. The default mode is `shared`.

The `shared` mode corresponds to the NVIDIA DEFAULT compute mode. The `exclusive_process` mode corresponds to the NVIDIA EXCLUSIVE_PROCESS compute mode.

mps=yes[,shared] | per_socket[,shared] | per_gpu[,shared] | no

Enables or disables the NVIDIA Multi-Process Service (MPS) for the GPUs that are allocated to the job. Using MPS effectively causes the EXCLUSIVE_PROCESS mode to behave like the DEFAULT mode for all MPS clients. MPS always allows multiple clients to use the GPU through the MPS server.

MPS is useful for both shared and exclusive process GPUs, and allows more efficient sharing of GPU resources and better GPU utilization. See the NVIDIA documentation for more information and limitations.

When using MPS, use the EXCLUSIVE_PROCESS mode to ensure that only a single MPS server is using the GPU, which provides additional insurance that the MPS server is the single point of arbitration between all CUDA process for that GPU.

You can also enable MPS daemon sharing by adding the **share** keyword with a comma and no space (for example, `mps=yes , shared` enables MPS daemon sharing on the host). If sharing is enabled, all jobs that are submitted by the same user with the same resource requirements share the same MPS daemon on the host, socket, or GPU.

LSF starts MPS daemons on a per-host, per-socket, or per-GPU basis depending on value of the **mps** keyword:

- If `mps=yes` is set, LSF starts one MPS daemon per host per job.
When `share` is enabled (that is, if `mps=yes , shared` is set), LSF starts one MPS daemon per host for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon on the host.
- If `mps=per_socket` is set, LSF starts one MPS daemon per socket per job. When enabled with `share` (that is, if `mps=per_socket , shared` is set), LSF starts one MPS daemon per socket for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon for the socket.
- If `mps=per_gpu` is set, LSF starts one MPS daemon per GPU per job. When enabled with `share` (that is, if `mps=per_gpu , shared` is set), LSF starts one MPS daemon per GPU for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon for the GPU.

Note: Using EXCLUSIVE_THREAD mode with MPS is not supported and might cause unexpected behavior.

j_exclusive=yes | no

Specifies whether the allocated GPUs can be used by other jobs. When the mode is set to `exclusive_process`, the `j_exclusive=yes` option is set automatically.

aff=yes | no

Specifies whether to enforce strict GPU-CPU affinity binding. If set to `no`, LSF relaxes GPU affinity while maintaining CPU affinity. By default, `aff=yes` is set to maintain strict GPU-CPU affinity binding.

The **LSB_GPU_NEW_SYNTAX=extend** or **LSB_GPU_NEW_SYNTAX=Y** parameter must be defined in the `lsf.conf` file to enable the `-gpu` option and **GPU_REQ** parameter syntax. When the **LSB_GPU_NEW_SYNTAX=extend** parameter is set, you cannot use the GPU resources `ngpus_shared`, `ngpus_excl_t` and `ngpus_excl_p` as job resource requirements.

If the **GPU_REQ_MERGE** parameter is defined as `Y` or `y` in the `lsb.params` file and a GPU requirement is specified at multiple levels (at least two of the default cluster, queue, application profile, or job level requirements), each option of the GPU requirement is merged separately. Job level overrides application level, which overrides queue level, which overrides the default cluster GPU requirement. For example, if the mode option of the GPU requirement is defined on the `-gpu` option, and the `mps` option is defined in the queue, the mode of job level and the `mps` value of queue is used.

If the **GPU_REQ_MERGE** parameter is not defined as `Y` or `y` in the `lsb.params` file and a GPU requirement is specified at multiple levels (at least two of the default cluster, queue, application profile, or job level requirements), the entire GPU requirement string is replaced. The entire job level GPU requirement string overrides application level, which overrides queue level, which overrides the default GPU requirement.

The **esub** parameter **LSB_SUB4_GPU_REQ** modifies the value of the `-gpu` option.

LSF selects the GPU that meets the topology requirement first. If the GPU mode of the selected GPU is not the requested mode, LSF changes the GPU to the requested mode. For example, if LSF allocates an `exclusive_process` GPU to a job that needs a shared GPU, LSF changes the GPU mode to `shared` before the job starts and then changes the mode back to `exclusive_process` when the job finishes.

The GPU requirements are converted to `rusage` resource requirements for the job. For example, `num=2` is converted to `rusage[ngpus_physical=2]`. Use the **bjobs**, **bhist**, and **bacct** commands to see the merged resource requirement.

There might be complex GPU requirements that the **bsub -gpu** option and **GPU_REQ** parameter syntax cannot cover, including compound GPU requirements (for different GPU requirements for jobs on different hosts, or for different parts of a parallel job) and alternate GPU requirements (if more than one set of GPU requirements might be acceptable for a job to run). For complex GPU requirements, use the **bsub -R** command option, or the **RES_REQ** parameter in the `lsb.applications` or `lsb.queues` file to define the resource requirement string.

Note: You can define the mode, `j_exclusive`, and `mps` options only with the `-gpu` option, the **LSB_GPU_REQ** parameter in the `lsf.conf` file, or the **GPU_REQ** parameter in the `lsb.queues` or `lsb.applications` files. You cannot use these options with the `rusage` resource requirement string in the **bsub -R** command option or the **RES_REQ** parameter in the `lsb.queues` or `lsb.applications` files.

Default

Not defined

See also

- **LSB_GPU_REQ**
- **LSB_GPU_NEW_SYNTAX**
- **bsub -gpu**

HOST_POST_EXEC

Syntax

HOST_POST_EXEC=*command*

Description

Enables host-based post-execution processing at the application level. The **HOST_POST_EXEC** command runs on all execution hosts after the job finishes. If job based post-execution **POST_EXEC** was defined at the queue-level/application-level/job-level, the **HOST_POST_EXEC** command runs after **POST_EXEC** of any level.

Host-based post-execution commands can be configured at the queue and application level, and run in the following order:

1. The application-level command
2. The queue-level command.

The supported command rule is the same as the existing **POST_EXEC** for the queue section. See the **POST_EXEC** topic for details.

Note:

The host-based post-execution command cannot be executed on Windows platforms. This parameter cannot be used to configure job-based post-execution processing.

Default

Not defined.

HOST_PRE_EXEC

Syntax

HOST_PRE_EXEC=*command*

Description

Enables host-based pre-execution processing at the application level. The **HOST_PRE_EXEC** command runs on all execution hosts before the job starts. If job based pre-execution **PRE_EXEC** was defined at the queue-level/application-level/job-level, the **HOST_PRE_EXEC** command runs before **PRE_EXEC** of any level.

Host-based pre-execution commands can be configured at the queue and application level, and run in the following order:

1. The queue-level command
2. The application-level command.

The supported command rule is the same as the existing **PRE_EXEC** for the queue section. See the **PRE_EXEC** topic for details.

Note:

The host-based pre-execution command cannot be executed on Windows platforms. This parameter cannot be used to configure job-based pre-execution processing.

Default

Not defined.

JOB_CWD

Syntax

`JOB_CWD=directory`

Description

Current working directory (CWD) for the job in the application profile. The path can be absolute or relative to the submission directory. The path can include the following dynamic patterns (which are case sensitive):

- %J - job ID
- %JG - job group (if not specified, it will be ignored)
- %I - job index (default value is 0)
- %EJ - execution job ID
- %EI - execution job index
- %P - project name
- %U - user name
- %G - user group
- %H - first execution host name

Unsupported patterns are treated as text.

If this parameter is changed, then any newly submitted jobs with the **-app** option will use the new value for CWD if **bsub -cwd** is not defined.

JOB_CWD supports all LSF path conventions such as UNIX, UNC and Windows formats. In the mixed UNIX /Windows cluster it can be specified with one value for UNIX and another value for Windows separated by a pipe character (|).

```
JOB_CWD=unix_path|windows_path
```

The first part of the path must be for UNIX and the second part must be for Windows. Both paths must be full paths.

Default

Not defined.

JOB_CWD_TTL

Syntax

`JOB_CWD_TTL=hours`

Description

Specifies the time-to-live (TTL) for the current working directory (CWD) of a job. LSF cleans created CWD directories after a job finishes based on the TTL value. LSF deletes the CWD for the job if LSF created that directory for the job. The following options are available:

- 0 - sbatchd deletes CWD when all process related to the job finish.
- 2147483647 - Never delete the CWD for a job.
- 1 to 2147483646 - Delete the CWD for a job after the timeout expires.

The system checks the directory list every 5 minutes with regards to cleaning and deletes only the last directory of the path to avoid conflicts when multiple jobs share some parent directories. TTL will be

calculated after the post-exec script finishes. When LSF (sbatchd) starts, it checks the directory list file and deletes expired CWDs.

If the value for this parameter is not set in the application profile, LSF checks to see if it is set at the cluster-wide level. If neither is set, the default value is used.

Default

Not defined. The value of 2147483647 is used, meaning the CWD is not deleted.

JOB_INCLUDE_POSTPROC

Syntax

JOB_INCLUDE_POSTPROC=Y | N

Description

Specifies whether LSF includes the post-execution processing of the job as part of the job. When set to Y:

- Prevents a new job from starting on a host until post-execution processing is finished on that host
- Includes the CPU and run times of post-execution processing with the job CPU and run times
- sbatchd sends both job finish status (**DONE** or **EXIT**) and post-execution processing status (**POST_DONE** or **POST_ERR**) to mbatchd at the same time

The variable LSB_JOB_INCLUDE_POSTPROC in the user environment overrides the value of JOB_INCLUDE_POSTPROC in an application profile in `lsb.applications`. JOB_INCLUDE_POSTPROC in an application profile in `lsb.applications` overrides the value of JOB_INCLUDE_POSTPROC in `lsb.params`.

For CPU and memory affinity jobs, if JOB_INCLUDE_POSTPROC=Y, LSF does not release affinity resources until post-execution processing has finished, since slots are still occupied by the job during post-execution processing.

For SGI cpusets, if JOB_INCLUDE_POSTPROC=Y, LSF does not release the cpuset until post-execution processing has finished, even though post-execution processes are not attached to the cpuset.

Default

N. Post-execution processing is not included as part of the job, and a new job can start on the execution host before post-execution processing finishes.

JOB_POSTPROC_TIMEOUT

Syntax

JOB_POSTPROC_TIMEOUT=*minutes*

Description

Specifies a timeout in minutes for job post-execution processing. The specified timeout must be greater than zero

If post-execution processing takes longer than the timeout, **sbatchd** reports that post-execution has failed (POST_ERR status). On UNIX and Linux, it kills the entire process group of the job's pre-execution processes. On Windows, only the parent process of the pre-execution command is killed when the timeout expires, the child processes of the pre-execution command are not killed.

If **JOB_INCLUDE_POSTPROC**=Y, and **sbatchd** kills the post-execution processes because the timeout has been reached, the CPU time of the post-execution processing is set to 0, and the job's CPU time does not include the CPU time of post-execution processing.

JOB_POSTPROC_TIMEOUT defined in an application profile in `lsb.applications` overrides the value in `lsb.params`. **JOB_POSTPROC_TIMEOUT** cannot be defined in user environment.

When running host-based post execution processing, set **JOB_POSTPROC_TIMEOUT** to a value that gives the process enough time to run.

Default

Not defined. Post-execution processing does not time out.

JOB_PREPROC_TIMEOUT

Syntax

`JOB_PREPROC_TIMEOUT=minutes`

Description

Specify a timeout in minutes for job pre-execution processing. The specified timeout must be an integer greater than zero. If the job's pre-execution processing takes longer than the timeout, LSF kills the job's pre-execution processes, kills the job with a pre-defined exit value of 98, and then requeues the job to the head of the queue. However, if the number of pre-execution retries has reached the pre-execution retry threshold, LSF suspends the job with PSUSP status instead of requeuing it.

`JOB_PREPROC_TIMEOUT` defined in an application profile in `lsb.applications` overrides the value in `lsb.params`. `JOB_PREPROC_TIMEOUT` cannot be defined in the user environment.

On UNIX and Linux, **sbatchd** kills the entire process group of the job's pre-execution processes.

On Windows, only the parent process of the pre-execution command is killed when the timeout expires, the child processes of the pre-execution command are not killed.

Default

Not defined. Pre-execution processing does not time out. However, when running host-based pre-execution processing, you cannot use the infinite value or it will fail. You must configure a reasonable value.

JOB_SIZE_LIST

Syntax

`JOB_SIZE_LIST=default_size [size ...]`

Description

A list of job sizes (number of tasks) that are allowed on this application.

When submitting a job or modifying a pending job that requests a job size by using the `-n` or `-R` options for **bsub** and **bmod**, the requested job size must be a single fixed value that matches one of the values that **JOB_SIZE_LIST** specifies, which are the job sizes that are allowed on this application profile. LSF rejects the job if the requested job size is not in this list. In addition, when using **bswitch** to switch a pending job with a requested job size to another queue, the requested job size in the pending job must also match one of the values in **JOB_SIZE_LIST** for the new queue.

The first value in this list is the default job size, which is the assigned job size request if the job was submitted without requesting one. The remaining values are the other job sizes allowed in the queue, and may be defined in any order.

When defined in both a queue (`lsb.queues`) and an application profile, the job size request must satisfy both requirements. In addition, **JOB_SIZE_LIST** overrides any **TASKLIMIT** parameters defined at the same level. Job size requirements do not apply to queues and application profiles with no job size lists, nor do they apply to other levels of job submissions (that is, host level or cluster level job submissions).

Note: An exclusive job may allocate more slots on the host than is required by the tasks. For example, if `JOB_SIZE_LIST=8` and an exclusive job requesting `-n8` runs on a 16 slot host, all 16 slots are assigned to the job. The job runs as expected, since the 8 tasks specified for the job matches the job size list.

Valid values

A space-separated list of positive integers between 1 and 2147483646.

Default

Undefined

JOB_STARTER

Syntax

```
JOB_STARTER=starter [starter] ["%USRCMD"] [starter]
```

Description

Creates a specific environment for submitted jobs prior to execution. An application-level job starter overrides a queue-level job starter.

starter is any executable that can be used to start the job (i.e., can accept the job as an input argument). Optionally, additional strings can be specified.

By default, the user commands run after the job starter. A special string, `%USRCMD`, can be used to represent the position of the user's job in the job starter command line. The `%USRCMD` string and any additional commands must be enclosed in quotation marks (" ").

Example

```
JOB_STARTER=csh -c "%USRCMD;sleep 10"
```

In this case, if a user submits a job

```
bsub myjob arguments
```

the command that actually runs is:

```
csh -c "myjob arguments;sleep 10"
```

Default

Not defined. No job starter is used,

LOCAL_MAX_PREEEXEC_RETRY

Syntax

```
LOCAL_MAX_PREEEXEC_RETRY=integer
```

Description

The maximum number of times to attempt the pre-execution command of a job on the local cluster.

When this limit is reached, the default behavior of the job is defined by the **LOCAL_MAX_PREEEXEC_RETRY_ACTION** parameter in `lsb.params`, `lsb.queues`, or `lsb.applications`.

Valid values

$0 < \text{MAX_PREEXEC_RETRY} < \text{INFINIT_INT}$

INFINIT_INT is defined in `lsf.h`.

Default

Not defined. The number of preexec retry times is unlimited

See also

LOCAL_MAX_PREEXEC_RETRY_ACTION in `lsb.params`, `lsb.queues`, and `lsb.applications`.

LOCAL_MAX_PREEXEC_RETRY_ACTION**Syntax**

`LOCAL_MAX_PREEXEC_RETRY_ACTION=SUSPEND | EXIT`

Description

The default behavior of a job when it reaches the maximum number of times to attempt its pre-execution command on the local cluster (**LOCAL_MAX_PREEXEC_RETRY** in `lsb.params`, `lsb.queues`, or `lsb.applications`).

- If set to `SUSPEND`, the job is suspended and its status is set to `PSUSP`.
- If set to `EXIT`, the job exits and its status is set to `EXIT`. The job exits with the same exit code as the last pre-execution fail exit code.

This parameter is configured cluster-wide (`lsb.params`), at the queue level (`lsb.queues`), and at the application level (`lsb.applications`). The action specified in `lsb.applications` overrides `lsb.queues`, and `lsb.queues` overrides the `lsb.params` configuration.

Default

Not defined. If not defined in `lsb.queues` or `lsb.params`, the default action is `SUSPEND`.

See also

LOCAL_MAX_PREEXEC_RETRY in `lsb.params`, `lsb.queues`, and `lsb.applications`.

MAX_JOB_PREEMPT**Syntax**

`MAX_JOB_PREEMPT=integer`

Description

The maximum number of times a job can be preempted. Applies to queue-based preemption only.

Valid values

$0 < \text{MAX_JOB_PREEMPT} < \text{INFINIT_INT}$

INFINIT_INT is defined in `lsf.h`.

Default

Not defined. The number of preemption times is unlimited.

MAX_JOB_REQUEUE

Syntax

MAX_JOB_REQUEUE=*integer*

Description

The maximum number of times to requeue a job automatically.

Valid values

0 < MAX_JOB_REQUEUE < INFINIT_INT

INFINIT_INT is defined in `lsf.h`.

Default

Not defined. The number of requeue times is unlimited

MAX_PREEEXEC_RETRY

Syntax

MAX_PREEEXEC_RETRY=*integer*

Description

Use REMOTE_MAX_PREEEXEC_RETRY instead. This parameter is only maintained for backwards compatibility.

Job forwarding model for the LSF multicluster capability only. The maximum number of times to attempt the pre-execution command of a job from a remote cluster.

If the job's pre-execution command fails all attempts, the job is returned to the submission cluster.

Valid values

0 < MAX_PREEEXEC_RETRY < INFINIT_INT

INFINIT_INT is defined in `lsf.h`.

Default

5

MAX_TOTAL_TIME_PREEMPT

Syntax

MAX_TOTAL_TIME_PREEMPT=*integer*

Description

The accumulated preemption time in minutes after which a job cannot be preempted again, where *minutes* is wall-clock time, not normalized time.

Setting this parameter in `lsb.applications` overrides the parameter of the same name in `lsb.queues` and in `lsb.params`.

Valid values

Any positive integer greater than or equal to one (1)

Default

Unlimited

MEMLIMIT**Syntax**

MEMLIMIT=*integer*

Description

The per-process (soft) process resident set size limit for all of the processes belonging to a job running in the application profile.

Sets the maximum amount of physical memory (resident set size, RSS) that may be allocated to a process.

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

By default, jobs submitted to the application profile without a job-level memory limit are killed when the memory limit is reached. Application-level limits override any default limit specified in the queue, but must be less than the hard limit of the submission queue.

LSF has two methods of enforcing memory usage:

- OS Memory Limit Enforcement
- LSF Memory Limit Enforcement

OS memory limit enforcement

OS memory limit enforcement is the default MEMLIMIT behavior and does not require further configuration. OS enforcement usually allows the process to eventually run to completion. LSF passes MEMLIMIT to the OS, which uses it as a guide for the system scheduler and memory allocator. The system may allocate more memory to a process if there is a surplus. When memory is low, the system takes memory from and lowers the scheduling priority (re-nice) of a process that has exceeded its declared MEMLIMIT. Only available on systems that support RLIMIT_RSS for **setrlimit()**.

Not supported on:

- Sun Solaris 2.x
- Windows

LSF memory limit enforcement

To enable LSF memory limit enforcement, set LSB_MEMLIMIT_ENFORCE in `lsf.conf` to *y*. LSF memory limit enforcement explicitly sends a signal to kill a running process once it has allocated memory past MEMLIMIT.

You can also enable LSF memory limit enforcement by setting LSB_JOB_MEMLIMIT in `lsf.conf` to *y*. The difference between LSB_JOB_MEMLIMIT set to *y* and LSB_MEMLIMIT_ENFORCE set to *y* is that with LSB_JOB_MEMLIMIT, only the per-job memory limit enforced by LSF is enabled. The per-process memory limit enforced by the OS is disabled. With LSB_MEMLIMIT_ENFORCE set to *y*, both the per-job memory limit enforced by LSF and the per-process memory limit enforced by the OS are enabled.

Available for all systems on which LSF collects total memory usage.

Default

Unlimited

MEMLIMIT_TYPE

Syntax

MEMLIMIT_TYPE=JOB [PROCESS] [TASK]

MEMLIMIT_TYPE=PROCESS [JOB] [TASK]

MEMLIMIT_TYPE=TASK [PROCESS] [JOB]

Description

A memory limit is the maximum amount of memory a job is allowed to consume. Jobs that exceed the level are killed. You can specify different types of memory limits to enforce. Use any combination of JOB, PROCESS, and TASK.

By specifying a value in the application profile, you overwrite these three parameters: LSB_JOB_MEMLIMIT, LSB_MEMLIMIT_ENFORCE, LSF_HPC_EXTENSIONS (TASK_MEMLIMIT).

Note: A task list is a list in LSF that keeps track of the default resource requirements for different applications and task eligibility for remote execution.

- **PROCESS:** Applies a memory limit by OS process, which is enforced by the OS on the slave machine (where the job is running). When the memory allocated to one process of the job exceeds the memory limit, LSF kills the job.
- **TASK:** Applies a memory limit based on the task list file. It is enforced by LSF. LSF terminates the entire parallel job if any single task exceeds the limit setting for memory and swap limits.
- **JOB:** Applies a memory limit identified in a job and enforced by LSF. When the sum of the memory allocated to all processes of the job exceeds the memory limit, LSF kills the job.
- **PROCESS TASK:** Enables both process-level memory limit enforced by OS and task-level memory limit enforced by LSF.
- **PROCESS JOB:** Enables both process-level memory limit enforced by OS and job-level memory limit enforced by LSF.
- **TASK JOB:** Enables both task-level memory limit enforced by LSF and job-level memory limit enforced by LSF.
- **PROCESS TASK JOB:** Enables process-level memory limit enforced by OS, task-level memory limit enforced by LSF, and job-level memory limit enforced by LSF.

Default

Not defined. The memory limit-level is still controlled by LSF_HPC_EXTENSIONS=TASK_MEMLIMIT, LSB_JOB_MEMLIMIT, LSB_MEMLIMIT_ENFORCE

MIG

Syntax

MIG=*minutes*

Description

Enables automatic job migration and specifies the migration threshold for checkpointable or rerunnable jobs, in minutes.

LSF automatically migrates jobs that have been in the SSUSP state for more than the specified number of minutes. A value of 0 specifies that a suspended job is migrated immediately. The migration threshold applies to all jobs running on the host.

Job-level command line migration threshold overrides threshold configuration in application profile and queue. Application profile configuration overrides queue level configuration.

When a host migration threshold is specified, and is lower than the value for the job, the queue, or the application, the host value is used.

Members of a chunk job can be migrated. Chunk jobs in WAIT state are removed from the job chunk and put into PEND state.

Does not affect jobs with the that are forwarded to a remote cluster.

Default

Not defined. LSF does not migrate checkpointable or rerunnable jobs automatically.

NAME

Syntax

NAME=*string*

Description

Required. Unique name for the application profile.

Specify any ASCII string up to 60 characters long. You can use letters, digits, underscores (_), dashes (-), periods (.) or spaces in the name. The application profile name must be unique within the cluster.

Note: If you want to specify the ApplicationVersion in a JSDL file, include the version when you define the application profile name. Separate the name and version by a space, as shown in the following example:

```
NAME=myapp 1.0
```

Default

You must specify this parameter to define an application profile. LSF does not automatically assign a default application profile name.

NETWORK_REQ

Syntax

NETWORK_REQ="*network_res_req*"

network_res_req has the following syntax:

```
[type=sn_all | sn_single] [:protocol=protocol_name[(protocol_number)]  
[, protocol_name[(protocol_number)]] [:mode=US | IP] [:usage=dedicated | shared]  
[:instance=positive_integer]
```

Description

For LSF IBM Parallel Environment (PE) integration. Specifies the network resource requirements for a PE job.

If any network resource requirement is specified in the job, queue, or application profile, the job is treated as a PE job. PE jobs can only run on hosts where IBM PE **pnsd** daemon is running.

The network resource requirement string *network_res_req* has the same syntax as the **bsub -network** option.

The `-network bsub` option overrides the value of `NETWORK_REQ` defined in `lsb.queues` or `lsb.applications`. The value of `NETWORK_REQ` defined in `lsb.applications` overrides queue-level `NETWORK_REQ` defined in `lsb.queues`.

The following IBM LoadLeveler job command file options are not supported in LSF:

- `collective_groups`
- `imm_send_buffers`
- `rcxtblocks`

The following network resource requirement options are supported:

type=sn_all | sn_single

Specifies the adapter device type to use for message passing: either `sn_all` or `sn_single`.

sn_single

When used for switch adapters, specifies that all windows are on a single network

sn_all

Specifies that one or more windows are on each network, and that striped communication should be used over all available switch networks. The networks specified must be accessible by all hosts selected to run the PE job. See the *Parallel Environment Runtime Edition for AIX: Operation and Use guide* (SC23-6781-05) for more information about submitting jobs that use striping.

If mode is IP and type is specified as `sn_all` or `sn_single`, the job will only run on InfiniBand (IB) adapters (IPoIB). If mode is IP and type is not specified, the job will only run on Ethernet adapters (IPoEth). For IPoEth jobs, LSF ensures the job is running on hosts where **pnsd** is installed and running. For IPoIB jobs, LSF ensures the job is running on hosts where **pnsd** is installed and running, and that IB networks are up. Because IP jobs do not consume network windows, LSF does not check if all network windows are used up or if the network is already occupied by a dedicated PE job.

Equivalent to the PE `MP_EUIDEVICE` environment variable and `-euidevice` PE flag See the *Parallel Environment Runtime Edition for AIX: Operation and Use guide* (SC23-6781-05) for more information. Only `sn_all` or `sn_single` are supported by LSF. The other types supported by PE are not supported for LSF jobs.

protocol=protocol_name[(protocol_number)]

Network communication protocol for the PE job, indicating which message passing API is being used by the application. The following protocols are supported by LSF:

mpi

The application makes only MPI calls. This value applies to any MPI job regardless of the library that it was compiled with (PE MPI, MPICH2).

pami

The application makes only PAMI calls.

lapi

The application makes only LAPI calls.

shmem

The application makes only OpenSHMEM calls.

user_defined_parallel_api

The application makes only calls from a parallel API that you define. For example:
`protocol=myAPI` or `protocol=charm`.

The default value is `mpi`.

LSF also supports an optional *protocol_number* (for example, `mpi(2)`), which specifies the number of contexts (endpoints) per parallel API instance. The number must be a power of 2, but no greater than

128 (1, 2, 4, 8, 16, 32, 64, 128). LSF will pass the communication protocols to PE without any change. LSF will reserve network windows for each protocol.

When you specify multiple parallel API protocols, you cannot make calls to both LAPI and PAMI (`lapi`, `pami`) or LAPI and OpenSHMEM (`lapi`, `shmem`) in the same application. Protocols can be specified in any order.

See the `MP_MSG_API` and `MP_ENDPOINTS` environment variables and the `-msg_api` and `-endpoints` PE flags in the *Parallel Environment Runtime Edition for AIX: Operation and Use* guide (SC23-6781-05) for more information about the communication protocols that are supported by IBM PE.

mode=US | IP

The network communication system mode used by the communication specified communication protocol: US (User Space) or IP (Internet Protocol). A US job can only run with adapters that support user space communications, such as the IB adapter. IP jobs can run with either Ethernet adapters or IB adapters. When IP mode is specified, the instance number cannot be specified, and network usage must be unspecified or shared.

Each instance on the US mode requested by a task running on switch adapters requires an adapter window. For example, if a task requests both the MPI and LAPI protocols such that both protocol instances require US mode, two adapter windows will be used.

The default value is US.

usage=dedicated | shared

Specifies whether the adapter can be shared with tasks of other job steps: `dedicated` or `shared`. Multiple tasks of the same job can share one network even if usage is `dedicated`.

The default usage is `shared`.

instances=positive_integer

The number of parallel communication paths (windows) per task made available to the protocol on each network. The number actually used depends on the implementation of the protocol subsystem.

The default value is 1.

If the specified value is greater than `MAX_PROTOCOL_INSTANCES` in `lsb.params` or `lsb.queues`, LSF rejects the job.

`LSF_PE_NETWORK_NUM` must be defined to a non-zero value in `lsf.conf` for `NETWORK_REQ` to take effect. If `LSF_PE_NETWORK_NUM` is not defined or is set to 0, `NETWORK_REQ` is ignored with a warning message.

Example

The following network resource requirement string specifies that the requirements for an `sn_all` job (one or more windows are on each network, and striped communication should be used over all available switch networks). The PE job uses MPI API calls (protocol), runs in user-space network communication system mode, and requires 1 parallel communication path (window) per task.

```
NETWORK_REQ = "protocol=mpi:mode=us:instance=1:type=sn_all"
```

Default

No default value, but if you specify no value (`NETWORK_REQ=""`), the job uses the following: `protocol=mpi:mode=US:usage=shared:instance=1` in the application profile.

NICE

Syntax

`NICE=integer`

Description

Adjusts the UNIX scheduling priority at which jobs from the application execute.

A value of 0 (zero) maintains the default scheduling priority for UNIX interactive jobs. This value adjusts the run-time priorities for batch jobs to control their effect on other batch or interactive jobs. See the `nice(1)` manual page for more details.

On Windows, this value is mapped to Windows process priority classes as follows:

- `nice` ≥ 0 corresponds to a priority class of IDLE
- `nice` < 0 corresponds to a priority class of NORMAL

LSF on Windows does not support HIGH or REAL - TIME priority classes.

When set, this value overrides **NICE** set at the queue level in `lsb . queues`.

Default

Not defined.

NO_PREEMPT_INTERVAL

Specifies the number of minutes a preemptable job can run before it is preempted. If the uninterrupted run time of a preemptable job is longer than the specified time, it can be preempted.

Syntax

`NO_PREEMPT_INTERVAL=minutes`

The value of *minutes* is wall-clock time, not normalized time.

Description

The `NO_PREEMPT_INTERVAL=0` parameter allows immediate preemption of jobs as soon as they start or resume running.

For example, if a job A needs to preempt other candidate preemptable jobs (B, C, and D), the **NO_PREEMPT_INTERVAL** parameter determines which job is preempted:

- Run time of job B and job C is less than the **NO_PREEMPT_INTERVAL** parameter: job B and C are not preempted.
- Run time of job D is greater than or equal to the **NO_PREEMPT_INTERVAL** parameter: job D is preempted.

Setting this parameter in `lsb . applications` overrides the parameter of the same name in the `lsb . queues` and in `lsb . params` files.

Default

0

NO_PREEMPT_FINISH_TIME

Syntax

`NO_PREEMPT_FINISH_TIME=minutes | percentage`

Description

Prevents preemption of jobs that will finish within the specified number of minutes or the specified percentage of the estimated run time or run limit.

Specifies that jobs due to finish within the specified number of minutes or percentage of job duration should not be preempted, where *minutes* is wall-clock time, not normalized time. Percentage must be greater than 0 or less than 100% (between 1% and 99%).

For example, if the job run limit is 60 minutes and NO_PREEMPT_FINISH_TIME=10%, the job cannot be preempted after it runs 54 minutes or longer.

If you specify percentage for NO_PREEMPT_FINISH_TIME, requires a run time (**bsub -We** or ESTIMATED_RUNTIME in `lsb.applications`), or run limit to be specified for the job (**bsub -W**, or RUNLIMIT in `lsb.queues`, or RUNLIMIT in `lsb.applications`)

NO_PREEMPT_RUN_TIME

Syntax

NO_PREEMPT_RUN_TIME=*minutes* | *percentage*

Description

Prevents preemption of jobs that have been running for the specified number of minutes or the specified percentage of the estimated run time or run limit.

Specifies that jobs that have been running for the specified number of minutes or longer should not be preempted, where *minutes* is wall-clock time, not normalized time. Percentage must be greater than 0 or less than 100% (between 1% and 99%).

For example, if the job run limit is 60 minutes and NO_PREEMPT_RUN_TIME=50%, the job cannot be preempted after it running 30 minutes or longer.

If you specify percentage for NO_PREEMPT_RUN_TIME, requires a run time (**bsub -We** or ESTIMATED_RUNTIME in `lsb.applications`), or run limit to be specified for the job (**bsub -W**, or RUNLIMIT in `lsb.queues`, or RUNLIMIT in `lsb.applications`)

PEND_TIME_LIMIT

Syntax

PEND_TIME_LIMIT=[*hour*:]*minute*

Description

Specifies the pending time limit for a job.

LSF sends the application-level pending time limit configuration to IBM Spectrum LSF RTM, which handles the alarm and triggered actions such as user notification (for example, notifying the user that submitted the job and the LSF administrator) and job control actions (for example, killing the job). IBM Spectrum LSF RTM compares the job's pending time to the pending time limit, and if the job is pending for longer than this specified time limit, IBM Spectrum LSF RTM triggers the alarm and actions. This parameter works without IBM Spectrum LSF RTM, but LSF does not take any other alarm actions.

In the job forwarding model for the LSF multicluster capability, the job's pending time limit is ignored in the execution cluster, while the submission cluster merges the job's queue-, application-, and job-level pending time limit according to local settings.

The pending time limit is in the form of [*hour*:]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The job-level pending time limit (**bsub -ptl**) overrides the application-level limit specified here, and the application-level limit overrides the queue-level limit (**PEND_TIME_LIMIT** in `lsb.queues`).

Default

Not defined.

PERSISTENT_HOST_ORDER

Syntax

PERSISTENT_HOST_ORDER=Y | yes | N | no

Description

Applies when migrating parallel jobs in the LSF multicluster capability. Setting

PERSISTENT_HOST_ORDER=Y ensures that jobs are restarted on hosts based on alphabetical names of the hosts, preventing them from being restarted on the same hosts that they ran on before migration.

Default

PERSISTENT_HOST_ORDER=N. Migrated jobs in LSF multicluster capability could run on the same hosts that they ran on before.

PLAN

For use when the **ALLOCATION_PLANNER** parameter is enabled. Used to identify the jobs that are candidates for planning.

Syntax

PLAN = Y | N | "<key>[value] ..."

Description

LSF requires that the **ALLOCATION_PLANNER** parameter is enabled in order to use **PLAN=Y**.

Also defined at the cluster and queue levels. The precedence is: application, queue, global. For example, application level setting overrides the queue level setting.

The following key-value pairs are supported:

key	value	Default	Description
DELAY	positive integer	-	Number of minutes to delay before considering making a plan for a job following the job's submission time.
MAX_JOBS	positive integer	-	Maximum number of jobs that can have a plan concurrently.

Note:

The **PLAN** parameter replaces the existing **SLOT_RESERVE** parameter and **RESOURCE_RESERVE** parameter when the **ALLOCATION_PLANNER** parameter is enabled.

Default

N

POST_EXEC

Syntax

POST_EXEC=command

Description

Enables post-execution processing at the application level. The **POST_EXEC** command runs on the execution host after the job finishes. Post-execution commands can be configured at the job, application, and queue levels.

If both application-level (**POST_EXEC** in `lsb.applications`) and job-level post-execution commands are specified, job level post-execution overrides application-level post-execution commands. Queue-level post-execution commands (**POST_EXEC** in `lsb.queues`) run after application-level post-execution and job-level post-execution commands.

The **POST_EXEC** command uses the same environment variable values as the job, and runs under the user account of the user who submits the job.

When a job exits with one of the application profile's **REQUEUE_EXIT_VALUES**, LSF requeues the job and sets the environment variable **LSB_JOBPEND**. The post-execution command runs after the requeued job finishes.

When the post-execution command is run, the environment variable `LSB_JOBEXIT_STAT` is set to the exit status of the job. If the execution environment for the job cannot be set up, `LSB_JOBEXIT_STAT` is set to 0 (zero).

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for `%J` (*job_ID*) and `%I` (*index_ID*).

For UNIX:

- The pre- and post-execution commands run in the `/tmp` directory under `/bin/sh -c`, which allows the use of shell features in the commands. The following example shows valid configuration lines:

```
PRE_EXEC= /usr/share/lsf/misc/testq_pre >> /tmp/pre.out
```

```
POST_EXEC= /usr/share/lsf/misc/testq_post | grep -v "Hey!"
```

- LSF sets the **PATH** environment variable to

```
PATH="/bin /usr/bin /sbin /usr/sbin"
```

- The stdin, stdout, and stderr are set to `/dev/null`
- To allow UNIX users to define their own post-execution commands, an LSF administrator specifies the environment variable `$USER_POSTEXEC` as the **POST_EXEC** command. A user then defines the post-execution command:

```
setenv USER_POSTEXEC /path_name
```

Note: The path name for the post-execution command must be an absolute path. This parameter cannot be used to configure host-based post-execution processing.

For Windows:

- The pre- and post-execution commands run under `cmd.exe /c`
- The standard input, standard output, and standard error are set to `NULL`
- The **PATH** is determined by the setup of the LSF Service

Note:

For post-execution commands that execute on a Windows Server 2003, x64 Edition platform, users must have read and execute privileges for `cmd.exe`.

Default

Not defined. No post-execution commands are associated with the application profile.

PREEMPT_DELAY

Syntax

PREEMPT_DELAY=*seconds*

Description

Preemptive jobs will wait the specified number of seconds from the submission time before preempting any low priority preemptable jobs. During the grace period, preemption will not be triggered, but the job can be scheduled and dispatched by other scheduling policies.

This feature can provide flexibility to tune the system to reduce the number of preemptions. It is useful to get better performance and job throughput. When the low priority jobs are short, if high priority jobs can wait a while for the low priority jobs to finish, preemption can be avoided and cluster performance is improved. If the job is still pending after the grace period has expired, the preemption will be triggered.

The waiting time is for preemptive jobs in the pending status only. It will not impact the preemptive jobs that are suspended.

The time is counted from the submission time of the jobs. The submission time means the time **mbatchd** accepts a job, which includes newly submitted jobs, restarted jobs (by **brestart**) or forwarded jobs from a remote cluster.

When the preemptive job is waiting, the pending reason is:

The preemptive job is allowing a grace period before preemption.

If you use an older version of **bjobs**, the pending reason is:

Unknown pending reason code <6701>;

The parameter is defined in `lsb.params`, `lsb.queues` (overrides `lsb.params`), and `lsb.applications` (overrides both `lsb.params` and `lsb.queues`).

Run **badmin reconfig** to make your changes take effect.

Default

Not defined (if the parameter is not defined anywhere, preemption is immediate).

PRE_EXEC

Syntax

PRE_EXEC=*command*

Description

Enables pre-execution processing at the application level. The **PRE_EXEC** command runs on the execution host before the job starts. If the **PRE_EXEC** command exits with a non-zero exit code, LSF requeues the job to the front of the queue.

Pre-execution commands can be configured at the application, queue, and job levels and run in the following order:

1. The queue-level command
2. The application-level or job-level command. If you specify a command at both the application and job levels, the job-level command overrides the application-level command; the application-level command is ignored.

The **PRE_EXEC** command uses the same environment variable values as the job, and runs under the user account of the user who submits the job.

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

For UNIX:

- The pre- and post-execution commands run in the /tmp directory under /bin/sh -c, which allows the use of shell features in the commands. The following example shows valid configuration lines:

```
PRE_EXEC= /usr/share/lsf/misc/testq_pre >> /tmp/pre.out
```

```
POST_EXEC= /usr/share/lsf/misc/testq_post | grep -v "Hey!"
```

- LSF sets the **PATH** environment variable to

```
PATH="/bin /usr/bin /sbin /usr/sbin"
```

- The stdin, stdout, and stderr are set to /dev/null

For Windows:

- The pre- and post-execution commands run under cmd.exe /c
- The standard input, standard output, and standard error are set to NULL
- The **PATH** is determined by the setup of the LSF Service

Note:

For pre-execution commands that execute on a Windows Server 2003, x64 Edition platform, users must have read and execute privileges for cmd.exe. This parameter cannot be used to configure host-based pre-execution processing.

Default

Not defined. No pre-execution commands are associated with the application profile.

PROCESSLIMIT

Syntax

```
PROCESSLIMIT=integer
```

Description

Limits the number of concurrent processes that can be part of a job.

By default, jobs submitted to the application profile without a job-level process limit are killed when the process limit is reached. Application-level limits override any default limit specified in the queue.

SIGINT, SIGTERM, and SIGKILL are sent to the job in sequence when the limit is reached.

Default

Unlimited

PRIORITY

Syntax

```
PRIORITY =integer
```

Description

Specifies a priority that is used as a factor when calculating the job priority for absolute job priority scheduling (APS).

Valid values

Specify an integer between 0 and 2147483646.

Default

Not defined.

If APS is enabled for users, user groups, or application profiles, the default value is 0.

RC_ACCOUNT

Assigns an account name (tag) to hosts borrowed through LSF resource connector, so that they cannot be used by other user groups, users, or jobs.

Syntax

RC_ACCOUNT=*account_name*

Description

When a job is submitted to an application profile with the **RC_ACCOUNT** parameter specified, hosts borrowed to run the job are tagged with the value of the **RC_ACCOUNT** parameter. The borrowed host cannot be used by other applications that have a different value for the **RC_ACCOUNT** parameter (or that don't have the **RC_ACCOUNT** parameter defined at all).

After the borrowed host joins the cluster, use the **lshosts -s** command to view the value of the **RC_ACCOUNT** parameter for the host.

Example

```
RC_ACCOUNT=project1
```

Default

No account defined for the application profile

REMOTE_MAX_PREEEXEC_RETRY

Syntax

REMOTE_MAX_PREEEXEC_RETRY=*integer*

Description

Job forwarding model for the LSF multicluster capability only. The maximum number of times to attempt the pre-execution command of a job from a remote cluster.

If the job's pre-execution command fails all attempts, the job is returned to the submission cluster.

Valid values

up to INFINIT_INT defined in `lsf.h`.

Default

5

REQUEUE_EXIT_VALUES

Syntax

```
REQUEUE_EXIT_VALUES=[exit_code ...] [EXCLUDE(exit_code ...)]
```

Description

Enables automatic job requeue and sets the LSB_EXIT_REQUEUE environment variable. Use spaces to separate multiple exit codes. Application-level exit values override queue-level values. Job-level exit values (**bsub -Q**) override application-level and queue-level values.

exit_code has the following form:

```
"[all] [-number ...] | [number ...]"
```

The reserved keyword all specifies all exit codes. Exit codes are typically between 0 and 255. Use a tilde (~) to exclude specified exit codes from the list.

Jobs are requeued to the head of the queue. The output from the failed run is not saved, and the user is not notified by LSF.

Define an exit code as EXCLUDE(*exit_code*) to enable exclusive job requeue, ensuring the job does not rerun on the samehost. Exclusive job requeue does not work for parallel jobs.

For jobs with the LSF multicluster capability forwarded to a remote execution cluster, the exit values specified in the submission cluster with the EXCLUDE keyword are treated as if they were non-exclusive.

You can also requeue a job if the job is terminated by a signal.

If a job is killed by a signal, the exit value is $128 + \text{signal_value}$. The sum of 128 and the signal value can be used as the exit code in the parameter REQUEUE_EXIT_VALUES.

For example, if you want a job to rerun if it is killed with a signal 9 (SIGKILL), the exit value would be $128 + 9 = 137$. You can configure the following requeue exit value to allow a job to be requeue if it was kill by signal 9:

```
REQUEUE_EXIT_VALUES=137
```

In Windows, if a job is killed by a signal, the exit value is *signal_value*. The signal value can be used as the exit code in the parameter REQUEUE_EXIT_VALUES.

For example, if you want to rerun a job after it was killed with a signal 7 (SIGKILL), the exit value would be 7. You can configure the following requeue exit value to allow a job to requeue after it was killed by signal 7:

```
REQUEUE_EXIT_VALUES=7
```

You can configure the following requeue exit value to allow a job to requeue for both Linux and Windows after it was killed:

```
REQUEUE_EXIT_VALUES=137 7
```

If mbatchd is restarted, it does not remember the previous hosts from which the job exited with an exclusive requeue exit code. In this situation, it is possible for a job to be dispatched to hosts on which the job has previously exited with an exclusive exit code.

You should configure REQUEUE_EXIT_VALUES for interruptible backfill queues (INTERRUPTIBLE_BACKFILL=*seconds*).

Example

```
REQUEUE_EXIT_VALUES=30 EXCLUDE(20)
```

means that jobs with exit code 30 are requeued, jobs with exit code 20 are requeued exclusively, and jobs with any other exit code are not requeued.

Default

Not defined. Jobs are not requeued.

RERUNNABLE

Syntax

RERUNNABLE=yes | no

Description

If yes, enables automatic job rerun (restart) for any job associated with the application profile.

Rerun is disabled when RERUNNABLE is set to no. The yes and no arguments are not case-sensitive.

Members of a chunk job can be rerunnable. If the execution host becomes unavailable, rerunnable chunk job members are removed from the job chunk and dispatched to a different execution host.

Job level rerun (**bsub -r**) overrides the RERUNNABLE value specified in the application profile, which overrides the queue specification. **bmod -rn** to make rerunnable jobs non-rerunnable overrides both the application profile and the queue.

Default

Not defined.

RES_REQ

Syntax

RES_REQ=*res_req*

Description

Resource requirements used to determine eligible hosts. Specify a resource requirement string as usual. The resource requirement string lets you specify conditions in a more flexible manner than using the load thresholds.

The following resource requirement sections are supported:

- select
- rusage
- order
- span
- same
- cu
- affinity

Resource requirement strings can be simple (applying to the entire job), compound (applying to the specified number of slots), or can contain alternative resources (alternatives between 2 or more simple and/or compound). When a compound resource requirement is set at the application-level, it will be ignored if any job-level resource requirements (simple or compound) are defined.

Compound and alternative resource requirements follow the same set of rules for determining how resource requirements are going to be merged between job, application, and queue level. In the event no job-level resource requirements are set, the compound or alternative application-level requirements interact with queue-level resource requirement strings in the following ways:

- When a compound resource requirement is set at the application level, it will be ignored if any job level resource requirements (simple or compound) are defined.

- If no queue-level resource requirement is defined or a compound or alternative queue-level resource requirement is defined, the application-level requirement is used.
- If a simple queue-level requirement is defined, the application-level and queue-level requirements combine as follows:

section	compound/alternative application and simple queue behavior
select	both levels satisfied; queue requirement applies to all terms
same	queue level ignored
order span	application-level section overwrites queue-level section (if a given level is present); queue requirement (if used) applies to all terms
rusage	<ul style="list-style-type: none"> – both levels merge – queue requirement if a job-based resource is applied to the first term, otherwise applies to all terms – if conflicts occur the application-level section overwrites the queue-level section. <p>For example: if the application-level requirement is $\text{num1} * \{ \text{rusage} [R1] \} + \text{num2} * \{ \text{rusage} [R2] \}$ and the queue-level requirement is $\text{rusage} [RQ]$ where RQ is a job resource, the merged requirement is $\text{num1} * \{ \text{rusage} [\text{merge} (R1, RQ)] \} + \text{num2} * \{ \text{rusage} [R2] \}$</p>

Compound or alternative resource requirements do not support the `cu` section, or the `||` operator within the `rusage` section.

Alternative resource strings use the `||` operator as a separator for each alternative resource.

Multiple `-R` strings cannot be used with multi-phase `rusage` resource requirements.

For internal load indices and duration, jobs are rejected if they specify resource reservation requirements at the job or application level that exceed the requirements specified in the queue.

By default, memory (`mem`) and swap (`swp`) limits in `select[]` and `rusage[]` sections are specified in MB. Use `LSF_UNIT_FOR_LIMITS` in `lsf.conf` to specify a larger unit for these limits (GB, TB, PB, or EB).

When `LSF_STRICT_RESREQ=Y` is configured in `lsf.conf`, resource requirement strings in `select` sections must conform to a more strict syntax. The strict resource requirement syntax only applies to the `select` section. It does not apply to the other resource requirement sections (`order`, `rusage`, `same`, `span`, `cu`, or `affinity`). When `LSF_STRICT_RESREQ=Y` in `lsf.conf`, LSF rejects resource requirement strings where an `rusage` section contains a non-consumable resource.

select section

For simple resource requirements, the `select` section defined at the application, queue, and job level must all be satisfied.

rusage section

The `rusage` section can specify additional requests. To do this, use the `OR (| |)` operator to separate additional `rusage` strings. The job-level `rusage` section takes precedence.

Note:

Compound resource requirements do not support use of the || operator within the component rusage simple resource requirements. Multiple rusage strings cannot be used with multi-phase rusage resource requirements.

When both job-level and application-level rusage sections are defined using simple resource requirement strings, the rusage section defined for the job overrides the rusage section defined in the application profile. The rusage definitions are merged, with the job-level rusage taking precedence. Any queue-level requirements are then merged with that result.

For example:

Application-level RES_REQ:

```
RES_REQ=rusage[mem=200:lic=1] ...
```

For the job submission:

```
bsub -R "rusage[mem=100]" ...
```

the resulting requirement for the job is

```
rusage[mem=100:lic=1]
```

where mem=100 specified by the job overrides mem=200 specified by the application profile. However, lic=1 from application profile is kept, since job does not specify it.

Application-level RES_REQ threshold:

```
RES_REQ = rusage[bwidth =2:threshold=5] ...
```

For the job submission:

```
bsub -R "rusage[bwidth =1:threshold=6]" ...
```

the resulting requirement for the job is

```
rusage[bwidth =1:threshold=6]
```

Application-level RES_REQ with decay and duration defined:

```
RES_REQ=rusage[mem=200:duration=20:decay=1] ...
```

For a job submission with no decay or duration:

```
bsub -R "rusage[mem=100]" ...
```

the resulting requirement for the job is:

```
rusage[mem=100:duration=20:decay=1]
```

Application-level duration and decay are merged with the job-level specification, and mem=100 for the job overrides mem=200 specified by the application profile. However, duration=20 and decay=1 from application profile are kept, since job does not specify them.

Application-level RES_REQ with resource reservation method:

```
RES_REQ=rusage[mem=200/host:duration=20:decay=1] ...
```

For a job submission with no decay or duration:

```
bsub -R' rusage[mem=100/task]' ...
```

the resulting requirement for the job is:

```
rusage[mem=100/task:duration=20:decay=1]
```

Application-level RES_REQ with multi-phase job-level rusage:

```
RES_REQ=rusage[mem=(200 150):duration=(10 10):decay=(1),swap=100] ...
```

For a multi-phase job submission:

```
bsub -app app_name -R "rusage[mem=(600 350):duration=(20 10):decay=(0 1)]" ...
```

the resulting requirement for the job is:

```
rusage[mem=(600 350):duration=(20 10):decay=(0 1),swap=100]
```

The job-level values for mem, duration and decay override the application-level values. However, swap=100 from the application profile is kept, since the job does not specify swap.

Application-level RES_REQ with multi-phase application-level rusage:

```
RES_REQ=rusage[mem=(200 150):duration=(10 10):decay=(1)] ...
```

For a job submission:

```
bsub -app app_name -R "rusage[mem=200:duration=15:decay=0]" ...
```

the resulting requirement for the job is:

```
rusage[mem=200:duration=15:decay=0]
```

Job-level values override the application-level multi-phase rusage string.

Note: The merged application-level and job-level rusage consumable resource requirements must satisfy any limits set by the parameter **RESRSV_LIMIT** in `lsb.queues`, or the job will be rejected.

order section

For simple resource requirements the `order` section defined at the job-level overrides any application-level order section. An application-level order section overrides queue-level specification. The `order` section defined at the application level is ignored if any resource requirements are specified at the job level. If the no resource requirements include an `order` section, the default order `r15s:pg` is used.

The command syntax is:

```
[!][-]resource_name [: [-]resource_name]
```

For example:

```
bsub -R "order[!ncpus:mem]" myjob
```

"!" only works with consumable resources because resources can be specified in the `rusage[]` section and their value may be changed in schedule cycle (for example, slot or memory). In LSF scheduler, slots under RUN, SSUSP, USUP and RSV may be freed in different scheduling phases. Therefore, the slot value may change in different scheduling cycles.

span section

For simple resource requirements the `span` section defined at the job-level overrides an application-level span section, which overrides a queue-level span section.

Note: Define `span[hosts=-1]` in the application profile or in **bsub -R** resource requirement string to disable the span section setting in the queue.

same section

For simple resource requirements all same sections defined at the job-level, application-level, and queue-level are combined before the job is dispatched.

cu section

For simple resource requirements the job-level cu section overrides the application-level, and the application-level cu section overrides the queue-level.

affinity section

For simple resource requirements the job-level affinity section overrides the application-level, and the application-level affinity section overrides the queue-level.

Default

```
select[type==local] order[r15s:pg]
```

If this parameter is defined and a host model or Boolean resource is specified, the default type is any.

RESIZABLE_JOBS

Syntax

```
RESIZABLE_JOBS = [Y|N|auto]
```

Description

N|n: The resizable job feature is disabled in the application profile. Under this setting, all jobs attached to this application profile are not resizable. All **bresize** and **bsub -ar** commands will be rejected with a proper error message.

Y|y: Resize is enabled in the application profile and all jobs belonging to the application are resizable by default. Under this setting, users can run **bresize** commands to cancel pending resource allocation requests for the job or release resources from an existing job allocation, or use **bsub** to submit an autoresizable job.

auto: All jobs belonging to the application will be autoresizable.

Resizable jobs must be submitted with an application profile that defines **RESIZABLE_JOBS** as either auto or Y. If an application defines **RESIZABLE_JOBS=auto**, but an administrator changes it to N and reconfigures LSF, jobs without the job-level auto resizable attribute are no longer autoresizable. For running jobs that are in the middle of the notification stage, LSF lets the current notification complete and stops scheduling. Changing **RESIZABLE_JOBS** configuration does not affect jobs with the job-level autoresizable attribute (This behavior is the same as for exclusive jobs, **bsub -x**, and the **EXCLUSIVE** parameter at the queue level).

Resizable jobs can have alternative and compound resource requirements. When using **bresize release** to release slots from compound resource requirements, you can only release slots represented by the last term of the compound resource requirement. To release slots in earlier terms, run **bresize release** repeatedly to release slots in subsequent last terms.

Default

If the parameter is undefined, the default value is N.

See also

RESIZABLE_JOBS in lsb.params

RESIZE_NOTIFY_CMD

Syntax

RESIZE_NOTIFY_CMD = *notification_command*

Description

Defines an executable command to be invoked on the first execution host of a job when a resize event occurs. The maximum length of notification command is 4 KB.

Default

Not defined. No resize notification command is invoked.

RESUME_CONTROL

Syntax

RESUME_CONTROL=*signal* | *command*

Remember: Unlike the JOB_CONTROLS parameter in `lsb.queues`, the RESUME_CONTROL parameter does not require square brackets ([]) around the action.

- *signal* is a UNIX signal name. The specified signal is sent to the job. The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the **kill -l** command.
- *command* specifies a `/bin/sh` command line to be invoked. Do not quote the command line inside an action definition. Do not specify a signal followed by an action that triggers the same signal. For example, do not specify `RESUME_CONTROL=bresume`. This causes a deadlock between the signal and the action.

Description

Changes the behavior of the RESUME action in LSF.

- The contents of the configuration line for the action are run with `/bin/sh -c` so you can use shell features in the command.
- The standard input, output, and error of the command are redirected to the NULL device, so you cannot tell directly whether the command runs correctly. The default null device on UNIX is `/dev/null`.
- The command is run as the user of the job.
- All environment variables set for the job are also set for the command action. The following additional environment variables are set:
 - `LSB_JOBPGIDS` — a list of current process group IDs of the job
 - `LSB_JOBPIIDS` — a list of current process IDs of the job
- If the command fails, LSF retains the original job status.

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for `%J` (*job_ID*) and `%I` (*index_ID*).

Default

- On UNIX, by default, RESUME sends SIGCONT.
- On Windows, actions equivalent to the UNIX signals have been implemented to do the default job control actions. Job control messages replace the SIGINT and SIGTERM signals, but only customized applications are able to process them.

RTASK_GONE_ACTION

Syntax

```
RTASK_GONE_ACTION="[KILLJOB_TASKDONE | KILLJOB_TASKEEXIT] [IGNORE_TASKCRASH]"
```

Description

Defines the actions LSF should take if it detects that a remote task of a parallel or distributed job is gone. This parameter only applies to the **blaunch** distributed application framework.

IGNORE_TASKCRASH

A remote task crashes. LSF does nothing. The job continues to launch the next task.

KILLJOB_TASKDONE

A remote task exits with zero value. LSF terminates all tasks in the job.

KILLJOB_TASKEEXIT

A remote task exits with non-zero value. LSF terminates all tasks in the job.

Environment variable

When defined in an application profile, the LSB_DJOB_RTASK_GONE_ACTION variable is set when running **bsub -app** for the specified application.

You can also use the environment variable LSB_DJOB_RTASK_GONE_ACTION to override the value set in the application profile.

Example

```
RTASK_GONE_ACTION="IGNORE_TASKCRASH KILLJOB_TASKEEXIT"
```

Default

Not defined. LSF does nothing.

RUNLIMIT

Syntax

```
RUNLIMIT=[hour:]minute[/host_name | /host_model]
```

Description

The default run limit. The name of a host or host model specifies the runtime normalization host to use.

By default, jobs that are in the RUN state for longer than the specified run limit are killed by LSF. You can optionally provide your own termination job action to override this default.

Jobs submitted with a job-level run limit (**bsub -w**) that is less than the run limit are killed when their job-level run limit is reached. Jobs submitted with a run limit greater than the maximum run limit are rejected. Application-level limits override any default limit specified in the queue.

Note: If you want to provide an estimated run time for scheduling purposes without killing jobs that exceed the estimate, define the ESTIMATED_RUNTIME parameter in the application profile, or submit the job with **-We** instead of a run limit.

The run limit is in the form of *[hour:]minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The run limit you specify is the normalized run time. This is done so that the job does approximately the same amount of processing, even if it is sent to host with a faster or slower CPU. Whenever a normalized run time is given, the actual time on the execution host is the specified time multiplied by the CPU factor of the normalization host then divided by the CPU factor of the execution host.

If `ABS_RUNLIMIT=Y` is defined in `lsb.params` or in the application profile, the runtime limit is not normalized by the host CPU factor. Absolute wall-clock run time is used for all jobs submitted to an application profile with a run limit configured.

Optionally, you can supply a host name or a host model name defined in LSF. You must insert `'/'` between the run limit and the host name or model name. (See **lsinfo(1)** to get host model information.)

If no host or host model is given, LSF uses the default runtime normalization host defined at the queue level (`DEFAULT_HOST_SPEC` in `lsb.queues`) if it has been configured; otherwise, LSF uses the default CPU time normalization host defined at the cluster level (`DEFAULT_HOST_SPEC` in `lsb.params`) if it has been configured; otherwise, the host with the largest CPU factor (the fastest host in the cluster).

For jobs with the LSF multicluster capability, if no other CPU time normalization host is defined and information about the submission host is not available, LSF uses the host with the largest CPU factor (the fastest host in the cluster).

Jobs submitted to a chunk job queue are not chunked if `RUNLIMIT` is greater than 30 minutes.

Default

Unlimited

RUNTIME

This parameter is deprecated. Use `ESTIMATED_RUNTIME` instead.

STACKLIMIT

Syntax

`STACKLIMIT=integer`

Description

The per-process (soft) stack segment size limit for all of the processes belonging to a job from this queue (see **getrlimit(2)**). Application-level limits override any default limit specified in the queue, but must be less than the hard limit of the submission queue.

By default, the limit is specified in KB. Use `LSF_UNIT_FOR_LIMITS` in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

Default

Unlimited

SUCCESS_EXIT_VALUES

Syntax

`SUCCESS_EXIT_VALUES=[exit_code ...]`

Description

Specifies exit values used by LSF to determine if job was done successfully. Use spaces to separate multiple exit codes. Job-level success exit values specified with the `LSB_SUCCESS_EXIT_VALUES` environment variable override the configuration in application profile.

Use `SUCCESS_EXIT_VALUES` for applications that successfully exit with non-zero values so that LSF does not interpret non-zero exit codes as job failure.

`exit_code` should be the value between 0 and 255. Use spaces to separate exit code values.

If both **SUCCESS_EXIT_VALUES** and **REQUEUE_EXIT_VALUES** are defined with the same exit code, **REQUEUE_EXIT_VALUES** will take precedence and the job will be set to PEND state and requeued.

Default

0

SUSPEND_CONTROL

Syntax

`SUSPEND_CONTROL=signal | command | CHPNT`

Remember: Unlike the `JOB_CONTROLS` parameter in `lsb.queues`, the `SUSPEND_CONTROL` parameter does not require square brackets (`[]`) around the action.

- `signal` is a UNIX signal name (for example, `SIGTSTP`). The specified signal is sent to the job. The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the `SIG` prefix) supported on your system, use the **kill -l** command.
- `command` specifies a `/bin/sh` command line to be invoked.
 - Do not quote the command line inside an action definition.
 - Do not specify a signal followed by an action that triggers the same signal. For example, do not specify `SUSPEND_CONTROL=bstop`. This causes a deadlock between the signal and the action.
- `CHKPNT` is a special action, which causes the system to checkpoint the job. The job is checkpointed and then stopped by sending the `SIGSTOP` signal to the job automatically.

Description

Changes the behavior of the `SUSPEND` action in LSF.

- The contents of the configuration line for the action are run with `/bin/sh -c` so you can use shell features in the command.
- The standard input, output, and error of the command are redirected to the `NULL` device, so you cannot tell directly whether the command runs correctly. The default null device on UNIX is `/dev/null`.
- The command is run as the user of the job.
- All environment variables set for the job are also set for the command action. The following additional environment variables are set:
 - `LSB_JOBPGIDS` - a list of current process group IDs of the job
 - `LSB_JOBPIIDS` - a list of current process IDs of the job
 - `LSB_SUSP_REASONS` - an integer representing a bitmap of suspending reasons as defined in `lsbatch.h`. The suspending reason can allow the command to take different actions based on the reason for suspending the job.
 - `LSB_SUSP_SUBREASONS` - an integer representing the load index that caused the job to be suspended
- If the command fails, LSF retains the original job status.

When the suspending reason `SUSP_LOAD_REASON` (suspended by load) is set in `LSB_SUSP_REASONS`, `LSB_SUSP_SUBREASONS` is set to one of the load index values defined in `lsf.h`.

Use `LSB_SUSP_REASONS` and `LSB_SUSP_SUBREASONS` together in your custom job control to determine the exact load threshold that caused a job to be suspended.

- If an additional action is necessary for the SUSPEND command, that action should also send the appropriate signal to the application. Otherwise, a job can continue to run even after being suspended by LSF. For example, `SUSPEND_CONTROL=bk111 $LSB_JOBPIIDS; command`

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

Default

- On UNIX, by default, SUSPEND sends SIGTSTP for parallel or interactive jobs and SIGSTOP for other jobs.
- On Windows, actions equivalent to the UNIX signals have been implemented to do the default job control actions. Job control messages replace the SIGINT and SIGTERM signals, but only customized applications are able to process them.

SWAPLIMIT

Syntax

SWAPLIMIT=*integer*

Description

Limits the amount of total virtual memory limit for the job.

This limit applies to the whole job, no matter how many processes the job may contain. Application-level limits override any default limit specified in the queue.

The action taken when a job exceeds its SWAPLIMIT or PROCESSLIMIT is to send SIGQUIT, SIGINT, SIGTERM, and SIGKILL in sequence. For CPULIMIT, SIGXCPU is sent before SIGINT, SIGTERM, and SIGKILL.

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in `lsf.conf` to specify a larger unit for the limit (MB, GB, TB, PB, or EB).

Default

Unlimited

TASKLIMIT

Syntax

TASKLIMIT=[*minimum_limit* [*default_limit*]] *maximum_limit*

Description

Note: **TASKLIMIT** replaces **PROCLIMIT** as of LSF 9.1.3.

Maximum number of tasks that can be allocated to a job. For parallel jobs, the maximum number of tasks that can be allocated to the job.

Queue level **TASKLIMIT** has the highest priority over application level **TASKLIMIT** and job level **TASKLIMIT**. Application level **TASKLIMIT** has higher priority than job level **TASKLIMIT**. Job-level limits must fall within the maximum and minimum limits of the application profile and the queue.

Note: If you also defined **JOB_SIZE_LIST** in the same application profile where you defined **TASKLIMIT**, the **TASKLIMIT** parameter is ignored.

Optionally specifies the minimum and default number of job tasks. All limits must be positive numbers greater than or equal to 1 that satisfy the following relationship:

$1 \leq \text{minimum} \leq \text{default} \leq \text{maximum}$

In the job forwarding model for the LSF multicluster capability, the local cluster considers the receiving queue's **TASKLIMIT** on remote clusters before forwarding jobs. If the receiving queue's **TASKLIMIT** definition in the remote cluster cannot satisfy the job's task requirements, the job is not forwarded to that remote queue.

Default

Unlimited, the default number of tasks is 1

TERMINATE_CONTROL

Syntax

TERMINATE_CONTROL=*signal* | *command* | CHPNT

Remember: Unlike the JOB_CONTROLS parameter in `lsb.queues`, the TERMINATE_CONTROL parameter does not require square brackets ([]) around the action.

- *signal* is a UNIX signal name (for example, SIGTERM). The specified signal is sent to the job. The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the **kill -l** command.
- *command* specifies a `/bin/sh` command line to be invoked.
 - Do not quote the command line inside an action definition.
 - Do not specify a signal followed by an action that triggers the same signal. For example, do not specify `TERMINATE_CONTROL=bkill`. This causes a deadlock between the signal and the action.
- CHPNT is a special action, which causes the system to checkpoint the job. The job is checkpointed and killed automatically.

Description

Changes the behavior of the TERMINATE action in LSF.

- The contents of the configuration line for the action are run with `/bin/sh -c` so you can use shell features in the command.
- The standard input, output, and error of the command are redirected to the NULL device, so you cannot tell directly whether the command runs correctly. The default null device on UNIX is `/dev/null`.
- The command is run as the user of the job.
- All environment variables set for the job are also set for the command action. The following additional environment variables are set:
 - LSB_JOBPGIDS — a list of current process group IDs of the job
 - LSB_JOBPIIDS — a list of current process IDs of the job

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

Default

- On UNIX, by default, TERMINATE sends SIGINT, SIGTERM and SIGKILL in that order.
- On Windows, actions equivalent to the UNIX signals have been implemented to do the default job control actions. Job control messages replace the SIGINT and SIGTERM signals, but only customized applications are able to process them. Termination is implemented by the **TerminateProcess()** system call.

THREADLIMIT

Syntax

THREADLIMIT=*integer*

Description

Limits the number of concurrent threads that can be part of a job. Exceeding the limit causes the job to terminate. The system sends the following signals in sequence to all processes belongs to the job: SIGINT, SIGTERM, and SIGKILL.

By default, jobs submitted to the queue without a job-level thread limit are killed when the thread limit is reached. Application-level limits override any default limit specified in the queue.

The limit must be a positive integer.

Default

Unlimited

USE_PAM_CREDS

Applies PAM limits to this application.

Syntax

USE_PAM_CREDS=y | n

Description

USE_PAM_CREDS is only supported on Linux systems.

If **USE_PAM_CREDS** is set to **y**, LSF applies PAM limits to an application when its job is dispatched to a Linux host using PAM. PAM limits are system resource limits defined in the `limits.conf` file. PAM limits override other limits. If the execution host does not have PAM configured and this parameter is enabled, the job fails.

For parallel jobs, **USE_PAM_CREDS** only takes effect on the first execution host.

Overrides **MEMLIMIT_TYPE=Process**.

Overridden (for CPU limit only) by **LSB_JOB_CPULIMIT=y**.

Overridden (for memory limits only) by **LSB_JOB_MEMLIMIT=y**.

Default

n. **USE_PAM_CREDS** is disabled.

WATCHDOG

Syntax

WATCHDOG=script[*file/path/to/script*] init[*init_delay*] period[*start_interval*]

Description

Enables LSF to use the watchdog feature to regularly run external scripts that check application data, logs, and other information. LSF can use these scripts to pass on the job information.

This parameter uses the following keywords:

script

Required. This keyword specifies the file path to the external watchdog script to check the application data and other information. This file must have the proper permissions for the job submission user to execute the script.

init

Optional. This keyword specifies the delay to start the watchdog script after the job starts, in seconds. Specify a number larger than 30 seconds. The default value is 60 seconds.

period

Optional. This keyword specifies the interval in which to start the watchdog script after the previous time that the watchdog script started, in seconds. Specify a number larger than 30 seconds. The default value is 60 seconds.

All job environment variables are available to the watchdog scripts. In addition, the following LSF job-level resource consumption environment variables are available to the watchdog scripts:

- **LSB_GPU_ALLOC_INFO**
- **LSB_JOB_AVG_MEM**
- **LSB_JOB_CPU_TIME**
- **LSB_JOB_MAX_MEM**
- **LSB_JOB_MEM**
- **LSB_JOB_NTHREAD**
- **LSB_JOB_PGIDS**
- **LSB_JOB_PIDS**
- **LSB_JOB_RUN_TIME**
- **LSB_JOB_SWAP**

Default

Not defined.

Automatic time-based configuration

Use if-else constructs and time expressions to define time windows in the file. Configuration defined within in a time window applies only during the specified time period; configuration defined outside of any time window applies at all times. After editing the file, run **badadmin reconfig** to reconfigure the cluster.

Time expressions in the file are evaluated by LSF every 10 minutes, based on the **mbatchd** daemon start time. When an expression evaluates true, LSF changes the configuration in real time, without restarting **mbatchd**, providing continuous system availability.

Time-based configuration also supports LSF multicluster capability configuration in terms of shared configuration for groups of clusters (using the **#include** parameter). You can include a common configuration file by using the time-based feature in local configuration files.

Example

```
Begin application
NAME=app1
#if time(16:00-18:00 EDT)
CPULIMIT=180/hostA
#else
CPULIMIT=60/hostA
#endif
End application
```

In this example, for two hours every day, the configuration is the following:

```
Begin application
NAME=app1
```

```
CPULIMIT=180/hostA
End application
```

The rest of the time, the configuration is the following:

```
Begin application
NAME=app1
CPULIMIT=60/hostA
End application
```

Specifying the time zone is optional. If you do not specify a time zone, LSF uses the local system time zone. LSF supports all standard time zone abbreviations.

lsb.events

The LSF batch event log file `lsb.events` is used to display LSF batch event history and for **mbatchd** failure recovery.

Whenever a host, job, or queue changes status, a record is appended to the event log file. The file is located in `LSB_SHAREDIR/cluster_name/logdir`, where **LSB_SHAREDIR** must be defined in the `lsf.conf` file, and `cluster_name` is the name of the LSF cluster, as returned by the **tlsid** command.

The **bhist** command searches the most current `lsb.events` file for its output.

lsb.events structure

The event log file is an ASCII file with one record per line. For the `lsb.events` file, the first line has the format `# history_seek_position>`, which indicates the file position of the first history event after log switch. For the `lsb.events.#` file, the first line has the format `# timestamp_most_recent_event`, which gives the timestamp of the most recent event in the file.

Limiting the size of lsb.events

Use the **MAX_JOB_NUM** parameter in the `lsb.params` file to set the maximum number of finished jobs whose events are to be stored in the `lsb.events` log file.

Once the limit is reached, **mbatchd** starts a new event log file. The old event log file is saved as `lsb.events.n`, with subsequent sequence number suffixes incremented by 1 each time a new log file is started. Event logging continues in the new `lsb.events` file.

Records and fields

The fields of a record are separated by blanks. The first string of an event record indicates its type. The following types of events are recorded:

- JOB_NEW
- JOB_FORWARD
- JOB_ACCEPT
- JOB_ACCEPTACK
- JOB_CHKPNT
- JOB_START
- JOB_START_ACCEPT
- JOB_STATUS
- JOB_SWITCH
- JOB_SWITCH2
- JOB_MOVE
- QUEUE_CTRL

lsb.events

- HOST_CTRL
- MBD_START
- MBD_DIE
- UNFULFILL
- LOAD_INDEX
- JOB_SIGACT
- MIG
- JOB_MODIFY2
- JOB_SIGNAL
- JOB_EXECUTE
- JOB_REQUEUE
- JOB_CLEAN
- JOB_EXCEPTION
- JOB_EXT_MSG
- JOB_ATTA_DATA
- JOB_CHUNK
- SBD_UNREPORTED_STATUS
- PRE_EXEC_START
- JOB_FORCE
- GRP_ADD
- GRP_MOD
- LOG_SWITCH
- JOB_RESIZE_NOTIFY_START
- JOB_RESIZE_NOTIFY_ACCEPT
- JOB_RESIZE_NOTIFY_DONE
- JOB_RESIZE_RELEASE
- JOB_RESIZE_CANCEL
- HOST_POWER_STATUS
- JOB_PROV_HOST

JOB_NEW

A new job has been submitted. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

userId (%d)

UNIX user ID of the submitter

options (%d)

Bit flags for job processing

numProcessors (%d)

Number of processors requested for execution

submitTime (%d)

Job submission time

beginTime (%d)

Start time – the job should be started on or after this time

termTime (%d)

Termination deadline – the job should be terminated by this time (%d)

sigValue (%d)

Signal value

chkpntPeriod (%d)

Checkpointing period

restartPid (%d)

Restart process ID

userName (%s)

User name

rLimitsSoft CPU time limit (%d), see **getrlimit(2)****rLimits**Soft file size limit (%d), see **getrlimit(2)****rLimits**Soft data segment size limit (%d), see **getrlimit(2)****rLimits**Soft stack segment size limit (%d), see **getrlimit(2)****rLimits**Soft core file size limit (%d), see **getrlimit(2)****rLimits**Soft memory size limit (%d), see **getrlimit(2)****rLimits**

Reserved (%d)

rLimits

Reserved (%d)

rLimits

Reserved (%d)

rLimitsSoft run time limit (%d), see **getrlimit(2)****rLimits**

Reserved (%d)

hostSpec (%s)

Model or host name for normalizing CPU time and run time

hostFactor (%f)

CPU factor of the above host

umask (%d)

File creation mask for this job

queue (%s)

Name of job queue to which the job was submitted

resReq (%s)

Resource requirements

fromHost (%s)

Submission host name

cwd (%s)

Current working directory (up to 4094 characters for UNIX or 255 characters for Windows)

chkpntDir (%s)

Checkpoint directory

inFile (%s)

Input file name (up to 4094 characters for UNIX or 255 characters for Windows)

outFile (%s)

Output file name (up to 4094 characters for UNIX or 255 characters for Windows)

errFile (%s)

Error output file name (up to 4094 characters for UNIX or 255 characters for Windows)

subHomeDir (%s)

Submitter's home directory

jobFile (%s)

Job file name

numAskedHosts (%d)

Number of candidate host names

askedHosts (%s)

List of names of candidate hosts for job dispatching

dependCond (%s)

Job dependency condition

preExecCmd (%s)

Job pre-execution command

jobName (%s)

Job name (up to 4094 characters)

command (%s)

Job command (up to 4094 characters for UNIX or 255 characters for Windows)

nxf (%d)

Number of files to transfer (%d)

xf (%s)

List of file transfer specifications

mailUser (%s)

Mail user name

projectName (%s)

Project name

niosPort (%d)

Callback port if batch interactive job

maxNumProcessors (%d)

Maximum number of processors

schedHostType (%s)

Execution host type

loginShell (%s)

Login shell

timeEvent (%d)

Time Event, for job dependency condition; specifies when time event ended

userGroup (%s)

User group

exceptList (%s)

Exception handlers for the job

options2 (%d)

Bit flags for job processing

idx (%d)

Job array index

inFileSpool (%s)

Spool input file (up to 4094 characters for UNIX or 255 characters for Windows)

commandSpool (%s)

Spool command file (up to 4094 characters for UNIX or 255 characters for Windows)

jobSpoolDir (%s)

Job spool directory (up to 4094 characters for UNIX or 255 characters for Windows)

userPriority (%d)

User priority

rsvId %s

Advance reservation ID; for example, "user12#0"

jobGroup (%s)

The job group under which the job runs

sla (%s)

SLA service class name under which the job runs

rLimits

Thread number limit

extsched (%s)

External scheduling options

warningAction (%s)

Job warning action

warningTimePeriod (%d)

Job warning time period in seconds

SLArunLimit (%d)

Absolute run time limit of the job for SLA service classes

licenseProject (%s)

IBM Spectrum LSF License Scheduler project name

options3 (%d)

Bit flags for job processing

app (%s)

Application profile name

postExecCmd (%s)

Post-execution command to run on the execution host after the job finishes

runtimeEstimation (%d)

Estimated run time for the job

requeueEValues (%s)

Job exit values for automatic job requeue

resizeNotifyCmd (%s)

Resize notification command to run on the first execution host to inform job of a resize event.

jobDescription (%s)

Job description (up to 4094 characters).

submitEXT

Submission extension field, reserved for internal use.

Num (%d)

Number of elements (key-value pairs) in the structure.

key (%s)

Reserved for internal use.

value (%s)

Reserved for internal use.

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

network (%s)

Network requirements for IBM Parallel Environment (PE) jobs.

cpu_frequency(%d)

CPU frequency at which the job runs.

options4 (%d)

Bit flags for job processing

nStinFile (%d)

(LSF Data Manager) The number of requested input files

stinFiles

(LSF Data Manager) List of input data requirement files requested. The list has the following elements:

options (%d)

Bit field that identifies whether the data requirement is an input file or a tag.

host (%s)

Source host of the input file. This field is empty if the data requirement is a tag.

name(%s)

Full path to the input data requirement file on the host. This field is empty if the data requirement is a tag.

hash (%s)

Hash key computed for the data requirement file at job submission time. This field is empty if the data requirement is a tag.

size (%lld)

Size of the data requirement file at job submission time in bytes.

modifyTime (%d)

Last modified time of the data requirement file at job submission time.

pendTimeLimit (%d)

Job-level pending time limit of the job, in seconds.

eligiblePendTimeLimit (%d)

Job-level eligible pending time limit of the job, in seconds.

JOB_FORWARD

A job has been forwarded to a remote cluster (IBM Spectrum LSF multicluster capability only).

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, older daemons and commands (pre-LSF Version 6.0) cannot recognize the `lsb.events` file format.

The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

numReserHosts (%d)

Number of reserved hosts in the remote cluster

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of `.hosts` listed in the `reserHosts` field.

cluster (%s)

Remote cluster name

reserHosts (%s)

List of names of the reserved hosts in the remote cluster

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format.

idx (%d)

Job array index

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

effectiveResReq (%s)

The runtime resource requirements used for the job.

ineligiblePendTime(%d)

Time in seconds that the job has been in the ineligible pending state.

JOB_ACCEPT

A job from a remote cluster has been accepted by this cluster. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID at the accepting cluster

remoteJid (%d)

Job ID at the submission cluster

cluster (%s)

Job submission cluster name

idx (%d)

Job array index

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

JOB_ACCEPTACK

Contains remote and local job ID mapping information. The default number for the ID is -1 (which means that this is not applicable to the job), and the default value for the cluster name is "" (empty string). The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

The ID number of the job at the execution cluster

idx (%d)

The job array index

jobRmtAttr (%d)

Remote job attributes from:

- Remote batch job on the submission side
- Lease job on the submission side
- Remote batch job on the execution side
- Lease job on the execution side
- Lease job re-synchronization during restart
- Remote batch job re-running on the execution cluster

srcCluster (%s)

The name of the submission cluster

srcJobId (%d)

The submission cluster job ID

dstCluster (%s)

The name of the execution cluster

dstJobId (%d)

The execution cluster job ID

JOB_CHKPNT

Contains job checkpoint information. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

The ID number of the job at the execution cluster

period (%d)

The new checkpointing period

jobPid (%d)

The process ID of the checkpointing process, which is a child sbatchd

ok (%d)

- 0 means the checkpoint started
- 1 means the checkpoint succeeded

flags (%d)

Checkpoint flags, see <lsf/lsbatch.h>:

- LSB_CHKPNT_KILL: Kill the process if checkpoint is successful
- LSB_CHKPNT_FORCE: Force checkpoint even if non-checkpointable conditions exist
- LSB_CHKPNT_MIG: Checkpoint for the purpose of migration

idx (%d)

Job array index (must be 0 in JOB_NEW)

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

JOB_START

A job has been dispatched.

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, older daemons and commands (pre-LSF Version 6.0) cannot recognize the `lsb.events` file format.

The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

jStatus (%d)

Job status, (4, indicating the RUN status of the job)

jobPid (%d)

Job process ID

jobPGid (%d)

Job process group ID

hostFactor (%f)

CPU factor of the first execution host

numExHosts (%d)

Number of processors used for execution

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of `.hosts` listed in the `execHosts` field.

execHosts (%s)

List of execution host names

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format.

queuePreCmd (%s)

Pre-execution command

queuePostCmd (%s)

Post-execution command

jFlags (%d)

Job processing flags

userGroup (%s)

User group name

idx (%d)

Job array index

additionalInfo (%s)

Placement information of HPC jobs

preemptBackfill (%d)

How long a backfilled job can run. Used for preemption backfill jobs.

jFlags2 (%d)

Job flags

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

effectiveResReq (%s)

The runtime resource requirements used for the job.

num_network (%d)

The number of the allocated network for IBM Parallel Environment (PE) jobs.

networkID (%s)

Network ID of the allocated network for IBM Parallel Environment (PE) jobs.

num_window (%d)

Number of allocated windows for IBM Parallel Environment (PE) jobs.

cpu_frequency(%d)

CPU frequency at which the job runs.

numAllocSlots(%d)

Number of allocated slots.

allocSlots(%s)

List of execution host names where the slots are allocated.

ineligiblePendTime(%d)

Time in seconds that the job has been in the ineligible pending state.

JOB_START_ACCEPT

A job has started on the execution host(s). The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

jobPid (%d)

Job process ID

jobPGid (%d)

Job process group ID

idx (%d)

Job array index

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

JOB_STATUS

The status of a job changed after dispatch. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

jStatus (%d)

New status, see <lsf/lSBATCH.h>

For JOB_STAT_EXIT (32) and JOB_STAT_DONE (64), host-based resource usage information is appended to the JOB_STATUS record in the fields numHostRusage and hostRusage.

reason (%d)

Pending or suspended reason code, see <lsf/lSBATCH.h>

subreasons (%d)

Pending or suspended subreason code, see <lsf/lSBATCH.h>

cpuTime (%f)

CPU time consumed so far

endTime (%d)

Job completion time

ru (%d)

Resource usage flag

lsfRusage (%s)

Resource usage statistics, see <lsf/lsf.h>

exitStatus (%d)

Exit status of the job, see <lsf/lSBATCH.h>

idx (%d)

Job array index

exitInfo (%d)

Job termination reason, see <lsf/lSBATCH.h>

duration4PreemptBackfill

How long a backfilled job can run. Used for preemption backfill jobs

numHostRusage(%d)

For a jStatus of JOB_STAT_EXIT (32) or JOB_STAT_DONE (64), this field contains the number of host-based resource usage entries (hostRusage) that follow. 0 unless LSF_HPC_EXTENSIONS="HOST_RUSAGE" is set in lsf.conf.

hostRusage

For a jStatus of JOB_STAT_EXIT (32) or JOB_STAT_DONE (64), these fields contain host-based resource usage information for the job for parallel jobs when LSF_HPC_EXTENSIONS="HOST_RUSAGE" is set in lsf.conf.

hostname (%s)

Name of the host.

mem(%d)

Total resident memory usage of all processes in the job running on this host.

swap(%d)

Total virtual memory usage of all processes in the job running on this host.

utime(%d)

User time used on this host.

stime(%d)

System time used on this host.

hHostExtendInfo(%d)

Number of following key-value pairs containing extended host information (PGIDs and PIDs). Set to 0 in lsb.events, lsb.acct, and lsb.stream files.

maxMem

Peak memory usage (in Mbytes)

avgMem

Average memory usage (in Mbytes)

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

ineligiblePendTime(%d)

Time in seconds that the job has been in the ineligible pending state. This is only recorded when the job is finished (DONE or EXIT) and not for any other changes in job status.

indexRangeCnt (%d)

The number of element ranges indicating successful signals

indexRangeStart1 (%d)

The start of the first index range.

indexRangeEnd1 (%d)

The end of the first index range.

indexRangeStep1 (%d)

The step of the first index range.

indexRangeStartN (%d)

The start of the last index range.

indexRangeEndN (%d)

The end of the last index range.

indexRangeStepN (%d)

The step of the last index range.

JOB_SWITCH

A job switched from one queue to another (**bswitch**). The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

userId (%d)

UNIX user ID of the user invoking the command

jobId (%d)

Job ID

queue (%s)

Target queue name

idx (%d)

Job array index. If it is -1, the `indexRangeCnt` takes effect.

userName (%s)

Name of the job submitter

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The target execution cluster job ID

dstCluster (%s)

The name of the execution cluster

rmtJobCtrlStage (%d)

The stage of remote job switch.

numRmtCtrlResult (%d)

The number of remote job switch record.

rmtCtrlResult

The record of each remote job switch session.

indexRangeCnt (%d)

The number of element ranges indicating successful signals

indexRangeStart1 (%d)

The start of the first index range.

indexRangeEnd1 (%d)

The end of the first index range.

indexRangeStep1 (%d)

The step of the first index range.

indexRangeStartN (%d)

The start of the last index range.

indexRangeEndN (%d)

The end of the last index range.

indexRangeStepN (%d)

The step of the last index range.

JOB_SWITCH2

A job array switched from one queue to another (**bswitch**). The fields are:

Version number (%s)

The version number

Event time (%d)

The time of the event

userId (%d)

UNIX user ID of the user invoking the command

jobId (%d)

Job ID

queue (%s)

Target queue name

userName (%s)

Name of the job submitter

indexRangeCnt (%s)

The number of element ranges indicating successful signals

indexRangeStart1 (%d)

The start of the first index range

indexRangeEnd1 (%d)

The end of the first index range

indexRangeStep1 (%d)

The step of the first index range

indexRangeStart2 (%d)

The start of the second index range

indexRangeEnd2 (%d)

The end of the second index range

indexRangeStep2 (%d)

The step of the second index range

indexRangeStartN (%d)

The start of the last index range

indexRangeEndN (%d)

The end of the last index range

indexRangeStepN (%d)

The step of the last index range

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

rmtCluster (%d)

The destination cluster to which the remote jobs belong

rmtJobCtrlId (%d)

Unique identifier for the remote job control session in the MultiCluster.

numSuccJobId (%d)

The number of jobs that were successful during this remote control operation.

succJobIdArray (%d)

Contains IDs for all the jobs that were successful during this remote control operation.

numFailJobId (%d)

The number of jobs which failed during this remote control session.

failJobIdArray (%d)

Contains IDs for all the jobs that failed during this remote control operation.

failReason (%d)

Contains the failure code and reason for each failed job in the failJobIdArray.

To prevent **JOB_SWITCH2** from getting too long, the number of index ranges is limited to 500 per **JOB_SWITCH2** event log. Therefore, if switching a large job array, several **JOB_SWITCH2** events may be generated.

JOB_MOVE

A job moved toward the top or bottom of its queue (**bbot** or **btot**). The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

userId (%d)

UNIX user ID of the user invoking the command

jobId (%d)

Job ID

position (%d)

Position number

base (%d)

Operation code, (TO_TOP or TO_BOTTOM), see <lsf/lSBATCH.h>

idx (%d)

Job array index

userName (%s)

Name of the job submitter

rmtJobCtrlStage (%d)

The stage of remote job move handling process.

numRmtCtrlResult (%d)

The stage of remote job move handling process.

rmtJobCtrlRecord

Remote job move result.

jobArrayIndex

Job array index.

numRmtCtrlResult2 (%d)

The number of records for remote job move handling.

rmtJobCtrlRecord2

Remote job move result.

QUEUE_CTRL

A job queue has been altered. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

opCode (%d)

Operation code), see <lsf/lsbatch.h>

queue (%s)

Queue name

userId (%d)

UNIX user ID of the user invoking the command

userName (%s)

Name of the user

ctrlComments (%s)

Administrator comment text from the -C option of **admin** queue control commands **qc**lose, **q**open, **q**act, and **q**inact

HOST_CTRL

A batch server host changed status. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

opCode (%d)

Operation code, see <lsf/lsbatch.h>

host (%s)

Host name

userId (%d)

UNIX user ID of the user invoking the command

userName (%s)

Name of the user

ctrlComments (%s)

Administrator comment text from the -C option of **badmin** host control commands **hclose** and **hopen**

MBD_START

The mbatchd has started. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

master (%s)

Master host name

cluster (%s)

cluster name

numHosts (%d)

Number of hosts in the cluster

numQueues (%d)

Number of queues in the cluster

MBD_DIE

The mbatchd died. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

master (%s)

Master host name

numRemoveJobs (%d)

Number of finished jobs that have been removed from the system and logged in the current event file

exitCode (%d)

Exit code from mbatchd

ctrlComments (%s)

Administrator comment text from the -C option of **badmin mbdrestart**

UNFULFILL

Actions that were not taken because the mbatchd was unable to contact the sbatchd on the job execution host. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

notSwitched (%d)

Not switched: the mbatchd has switched the job to a new queue, but the sbatchd has not been informed of the switch

sig (%d)

Signal: this signal has not been sent to the job

sig1 (%d)

Checkpoint signal: the job has not been sent this signal to checkpoint itself

sig1Flags (%d)

Checkpoint flags, see <lsf/lSBATCH.h>

chkPeriod (%d)

New checkpoint period for job

notModified (%s)

If set to true, then parameters for the job cannot be modified.

idx (%d)

Job array index

LOAD_INDEX

mbatchd restarted with these load index names (see `lsf.cluster(5)`). The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

nIdx (%d)

Number of index names

name (%s)

List of index names

JOB_SIGACT

An action on a job has been taken. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

period (%d)

Action period

pid (%d)

Process ID of the child sbatchd that initiated the action

jstatus (%d)

Job status

reasons (%d)

Job pending reasons

flags (%d)

Action flags, see <lsf/lSBATCH.h>

actStatus (%d)

Action status:

1: Action started

2: One action preempted other actions

3: Action succeeded

4: Action Failed

signalSymbol (%s)

Action name, accompanied by actFlags

idx (%d)

Job array index

MIG

A job has been migrated (**bmig**). The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

numAskedHosts (%d)

Number of candidate hosts for migration

askedHosts (%s)

List of names of candidate hosts

userId (%d)

UNIX user ID of the user invoking the command

idx (%d)

Job array index

userName (%s)

Name of the job submitter

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

JOB_MODIFY2

This is created when the mbatchd modifies a previously submitted job with **bmod**.

Version number (%s)

The version number

Event time (%d)

The time of the event

jobIdStr (%s)

Job ID

options (%d)

Bit flags for job modification options processing

options2 (%d)

Bit flags for job modification options processing

delOptions (%d)

Delete options for the options field

userId (%d)

UNIX user ID of the submitter

userName (%s)

User name

submitTime (%d)

Job submission time

umask (%d)

File creation mask for this job

numProcessors (%d)

Number of processors requested for execution. The value 2147483646 means the number of processors is undefined.

beginTime (%d)

Start time – the job should be started on or after this time

termTime (%d)

Termination deadline – the job should be terminated by this time

sigValue (%d)

Signal value

restartPid (%d)

Restart process ID for the original job

jobName (%s)

Job name (up to 4094 characters)

queue (%s)

Name of job queue to which the job was submitted

numAskedHosts (%d)

Number of candidate host names

askedHosts (%s)

List of names of candidate hosts for job dispatching; blank if the last field value is 0. If there is more than one host name, then each additional host name will be returned in its own field

resReq (%s)

Resource requirements

rLimits

Soft CPU time limit (%d), see **getrlimit(2)**

rLimits

Soft file size limit (%d), see **getrlimit(2)**

rLimits

Soft data segment size limit (%d), see **getrlimit(2)**

rLimits

Soft stack segment size limit (%d), see **getrlimit(2)**

rLimits

Soft core file size limit (%d), see **getrlimit(2)**

rLimits

Soft memory size limit (%d), see **getrlimit(2)**

rLimits

Reserved (%d)

rLimits

Reserved (%d)

rLimits

Reserved (%d)

rLimits

Soft run time limit (%d), see **getrlimit(2)**

rLimits

Reserved (%d)

hostSpec (%s)

Model or host name for normalizing CPU time and run time

dependCond (%s)

Job dependency condition

timeEvent (%d)

Time Event, for job dependency condition; specifies when time event ended

subHomeDir (%s)

Submitter's home directory

inFile (%s)

Input file name (up to 4094 characters for UNIX or 255 characters for Windows)

outFile (%s)

Output file name (up to 4094 characters for UNIX or 255 characters for Windows)

errFile (%s)

Error output file name (up to 4094 characters for UNIX or 255 characters for Windows)

command (%s)

Job command (up to 4094 characters for UNIX or 255 characters for Windows)

chkpntPeriod (%d)

Checkpointing period

chkpntDir (%s)

Checkpoint directory

nxf (%d)

Number of files to transfer

xf (%s)

List of file transfer specifications

jobFile (%s)

Job file name

fromHost (%s)

Submission host name

cwd (%s)

Current working directory (up to 4094 characters for UNIX or 255 characters for Windows)

preExecCmd (%s)

Job pre-execution command

mailUser (%s)

Mail user name

projectName (%s)

Project name

niosPort (%d)

Callback port if batch interactive job

maxNumProcessors (%d)

Maximum number of processors. The value 2147483646 means the maximum number of processors is undefined.

loginShell (%s)

Login shell

schedHostType (%s)

Execution host type

userGroup (%s)

User group

exceptList (%s)

Exception handlers for the job

delOptions2 (%d)

Delete options for the options2 field

inFileSpool (%s)

Spool input file (up to 4094 characters for UNIX or 255 characters for Windows)

commandSpool (%s)

Spool command file (up to 4094 characters for UNIX or 255 characters for Windows)

userPriority (%d)

User priority

rsvId %s

Advance reservation ID; for example, "user12#0"

extsched (%s)

External scheduling options

warningTimePeriod (%d)

Job warning time period in seconds

warningAction (%s)

Job warning action

jobGroup (%s)

The job group to which the job is attached

sla (%s)

SLA service class name that the job is to be attached to

licenseProject (%s)

IBM Spectrum LSF License Scheduler project name

options3 (%d)

Bit flags for job processing

delOption3 (%d)

Delete options for the options3 field

app (%s)

Application profile name

apsString (%s)

Absolute priority scheduling (APS) value set by administrator

postExecCmd (%s)

Post-execution command to run on the execution host after the job finishes

runtimeEstimation (%d)

Estimated run time for the job

requeueEValues (%s)

Job exit values for automatic job requeue

resizeNotifyCmd (%s)

Resize notification command to run on the first execution host to inform job of a resize event.

jobdescription (%s)

Job description (up to 4094 characters).

submitEXT

Submission extension field, reserved for internal use.

Num (%d)

Number of elements (key-value pairs) in the structure.

key (%s)

Reserved for internal use.

value (%s)

Reserved for internal use.

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

network (%s)

Network requirements for IBM Parallel Environment (PE) jobs.

cpu_frequency(%d)

CPU frequency at which the job runs.

options4 (%d)

Bit flags for job processing

nStinFile (%d)

(LSF Data Manager) The number of requested input files

stinFiles

(LSF Data Manager) List of input data requirement files requested. The list has the following elements:

options (%d)

Bit field that identifies whether the data requirement is an input file or a tag.

host (%s)

Source host of the input file. This field is empty if the data requirement is a tag.

name(%s)

Full path to the input data requirement file on the host. This field is empty if the data requirement is a tag.

hash (%s)

Hash key computed for the data requirement file at job submission time. This field is empty if the data requirement is a tag.

size (%lld)

Size of the data requirement file at job submission time in bytes.

modifyTime (%d)

Last modified time of the data requirement file at job submission time.

pendTimeLimit (%d)

Job-level pending time limit of the job, in seconds.

eligiblePendTimeLimit (%d)

Job-level eligible pending time limit of the job, in seconds.

dataGrp (%s)

Data group name.

numRmtCtrlResult2 (%s)

Number of remote job modification sessions that are generated by this modify request. JobIDs are recorded in an array index range format.

rmtCtrlResult2 (%s)

Remote job modification records

indexRangeCnt (%d)

The number of element ranges indicating successful signals

indexRangeStart1 (%d)

The start of the first index range.

indexRangeEnd1 (%d)

The end of the first index range.

indexRangeStep1 (%d)

The step of the first index range.

indexRangeStartN (%d)

The start of the last index range.

indexRangeEndN (%d)

The end of the last index range.

indexRangeStepN (%d)

The step of the last index range.

JOB_SIGNAL

This is created when a job is signaled with **kill** or deleted with **del**. The fields are in the order they appended:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

userId (%d)

UNIX user ID of the user invoking the command

runCount (%d)

Number of runs

signalSymbol (%s)

Signal name

idx (%d)

Job array index

userName (%s)

Name of the job submitter

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

indexRangeCnt (%d)

The number of element ranges indicating successful signals

indexRangeStart1 (%d)

The start of the first index range.

indexRangeEnd1 (%d)

The end of the first index range.

indexRangeStep1 (%d)

The step of the first index range.

indexRangeStartN (%d)

The start of the last index range.

indexRangeEndN (%d)

The end of the last index range.

indexRangeStepN (%d)

The step of the last index range.

jStatus (&d)

The job status.

JOB_EXECUTE

This is created when a job is actually running on an execution host. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

execUid (%d)

Mapped UNIX user ID on execution host

jobPGid (%d)

Job process group ID

execCwd (%s)

Current working directory job used on execution host (up to 4094 characters for UNIX or 255 characters for Windows)

execHome (%s)

Home directory job used on execution host

execUsername (%s)

Mapped user name on execution host

jobPid (%d)

Job process ID

idx (%d)

Job array index

additionalInfo (%s)

Placement information of HPC jobs

SLAscaledRunLimit (%d)

Run time limit for the job scaled by the execution host

execRusage

An internal field used by LSF.

Position

An internal field used by LSF.

duration4PreemptBackfill

How long a backfilled job can run; used for preemption backfill jobs

srcJobId (%d)

The submission cluster job ID

srcCluster (%s)

The name of the submission cluster

dstJobId (%d)

The execution cluster job ID

dstCluster (%s)

The name of the execution cluster

JOB_QUEUE

This is created when a job ended and requeued by mbatchd. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

idx (%d)

Job array index

JOB_CLEAN

This is created when a job is removed from the mbatchd memory. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

idx (%d)

Job array index

indexRangeCnt (%d)

The number of element ranges indicating successful signals

indexRangeStart1 (%d)

The start of the first index range.

indexRangeEnd1 (%d)

The end of the first index range.

indexRangeStep1 (%d)

The step of the first index range.

indexRangeStartN (%d)

The start of the last index range.

indexRangeEndN (%d)

The end of the last index range.

indexRangeStepN (%d)

The step of the last index range.

JOB_EXCEPTION

This is created when an exception condition is detected for a job. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

exceptMask (%d)

Exception Id

0x01: missed

0x02: overrun

0x04: underrun

0x08: abend

0x10: cantrun

0x20: hostfail

0x40: startfail

0x100:runtime_est_exceeded

actMask (%d)

Action Id

0x01: kill

0x02: alarm

0x04: rerun

0x08: setexcept

timeEvent (%d)

Time Event, for missed exception specifies when time event ended.

exceptInfo (%d)

Except Info, pending reason for missed or cantrun exception, the exit code of the job for the abend exception, otherwise 0.

idx (%d)

Job array index

JOB_EXT_MSG

An external message has been sent to a job. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

idx (%d)

Job array index

msgIdx (%d)

Index in the list

userId (%d)

Unique user ID of the user invoking the command

dataSize (%ld)

Size of the data if it has any, otherwise 0

postTime (%ld)

Message sending time

dataStatus (%d)

Status of the attached data

desc (%s)

Text description of the message

userName (%s)

Name of the author of the message

Flags (%d)

Used for internal flow control

JOB_ATTA_DATA

An update on the data status of a message for a job has been sent. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

idx (%d)

Job array index

msgIdx (%d)

Index in the list

dataSize (%ld)

Size of the data if it has any, otherwise 0

dataStatus (%d)

Status of the attached data

fileName (%s)

File name of the attached data

JOB_CHUNK

This is created when a job is inserted into a chunk.

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, older daemons and commands (pre-LSF Version 6.0) cannot recognize the `lsb.events` file format.

The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

membSize (%ld)

Size of array `membJobId`

membJobId (%ld)

Job IDs of jobs in the chunk

numExHosts (%ld)

Number of execution hosts

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of `.hosts` listed in the `execHosts` field.

execHosts (%s)

Execution host name array

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format.

SBD_UNREPORTED_STATUS

This is created when an unreported status change occurs. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

actPid (%d)

Acting processing ID

jobPid (%d)

Job process ID

jobPGid (%d)

Job process group ID

newStatus (%d)

New status of the job

reason (%d)

Pending or suspending reason code, see <lsf/lsbatch.h>

suspreason (%d)

Pending or suspending subreason code, see <lsf/lsbatch.h>

lsfRusage

The following fields contain resource usage information for the job (see **getrusage(2)**). If the value of some field is unavailable (due to job exit or the difference among the operating systems), -1 will be logged. Times are measured in seconds, and sizes are measured in KB.

ru_utime (%f)

User time used

ru_stime (%f)

System time used

ru_maxrss (%f)

Maximum shared text size

ru_ixrss (%f)

Integral of the shared text size over time (in KB seconds)

ru_ismrss (%f)

Integral of the shared memory size over time (valid only on Ultrix)

ru_idrss (%f)

Integral of the unshared data size over time

ru_isrss (%f)

Integral of the unshared stack size over time

ru_minflt (%f)

Number of page reclaims

ru_majflt (%f)

Number of page faults

ru_nswap (%f)

Number of times the process was swapped out

ru_inblock (%f)

Number of block input operations

ru_oublock (%f)

Number of block output operations

ru_ioch (%f)

Number of characters read and written (valid only on HP-UX)

ru_msgsnd (%f)

Number of System V IPC messages sent

ru_msgrcv (%f)

Number of messages received

ru_nsignals (%f)

Number of signals received

ru_nvcsw (%f)

Number of voluntary context switches

ru_nivcsw (%f)

Number of involuntary context switches

ru_exutime (%f)

Exact user time used (valid only on ConvexOS)

exitStatus (%d)

Exit status of the job, see <lsf/lSBATCH.h>

execCwd (%s)

Current working directory job used on execution host (up to 4094 characters for UNIX or 255 characters for Windows)

execHome (%s)

Home directory job used on execution host

execUsername (%s)

Mapped user name on execution host

msgId (%d)

ID of the message

actStatus (%d)

Action status

1: Action started

2: One action preempted other actions

3: Action succeeded

4: Action Failed

sigValue (%d)

Signal value

seq (%d)

Sequence status of the job

idx (%d)

Job array index

jRusage

The following fields contain resource usage information for the job. If the value of some field is unavailable (due to job exit or the difference among the operating systems), -1 will be logged. Times are measured in seconds, and sizes are measured in KB.

mem (%d)

Total resident memory usage in KB of all currently running processes in a given process group

swap (%d)

Totally virtual memory usage in KB of all currently running processes in given process groups

utime (%d)

Cumulative total user time in seconds

stime (%d)

Cumulative total system time in seconds

npids (%d)

Number of currently active process in given process groups. This entry has four sub-fields:

pid (%d)

Process ID of the child sbatchd that initiated the action

ppid (%d)

Parent process ID

pgid (%d)

Process group ID

jobId (%d)

Process Job ID

npgids (%d)

Number of currently active process groups

exitInfo (%d)

Job termination reason, see <lsf/lSBATCH.h>

PRE_EXEC_START

A pre-execution command has been started.

The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

jStatus (%d)

Job status, (4, indicating the RUN status of the job)

jobPid (%d)

Job process ID

jobPGid (%d)

Job process group ID

hostFactor (%f)

CPU factor of the first execution host

numExHosts (%d)

Number of processors used for execution

execHosts (%s)

List of execution host names

queuePreCmd (%s)

Pre-execution command

queuePostCmd (%s)

Post-execution command

jFlags (%d)

Job processing flags

userGroup (%s)

User group name

idx (%d)

Job array index

additionalInfo (%s)

Placement information of HPC jobs

effectiveResReq (%s)

The runtime resource requirements used for the job.

JOB_FORCE

A job has been forced to run with **brun**.

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

userId (%d)

UNIX user ID of the user invoking the command

idx (%d)

Job array index

options (%d)

Bit flags for job processing

numExecHosts (%ld)

Number of execution hosts

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of `.hosts` listed in the `execHosts` field.

execHosts (%s)

Execution host name array

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format.

userName (%s)

Name of the user

queue (%s)

Name of queue if a remote brun job ran; otherwise, this field is empty. For MultiCluster this is the name of the receive queue at the execution cluster.

GRP_ADD

This is created when a job group is added. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

userId (%d)

UNIX user ID of the job group owner

submitTime (%d)

Job submission time

userName (%s)

User name of the job group owner

depCond (%s)

Job dependency condition

timeEvent (%d)

Time Event, for job dependency condition; specifies when time event ended

groupSpec (%s)

Job group name

delOptions (%d)

Delete options for the options field

delOptions2 (%d)

Delete options for the options2 field

sla (%s)

SLA service class name that the job group is to be attached to

maxJLimit (%d)

Job group limit set by `bgadd -L`

groupType (%d)

Job group creation method:

- 0x01 - job group was created explicitly
- 0x02 - job group was created implicitly

GRP_MOD

This is created when a job group is modified. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

userId (%d)

UNIX user ID of the job group owner

submitTime (%d)

Job submission time

userName (%s)

User name of the job group owner

depCond (%s)

Job dependency condition

timeEvent (%d)

Time Event, for job dependency condition; specifies when time event ended

groupSpec (%s)

Job group name

delOptions (%d)

Delete options for the options field

delOptions2 (%d)

Delete options for the options2 field

sla (%s)

SLA service class name that the job group is to be attached to

maxJLimit (%d)

Job group limit set by **bgmod -L**

LOG_SWITCH

This is created when switching the event file `lsb.events`. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

JOB_RESIZE_NOTIFY_START

LSF logs this event when a resize (shrink or grow) request has been sent to the first execution host. The fields in order of occurrence are:

Version number (%s)

The version number.

Event time (%d)

The time of the event.

jobId (%d)

The job ID.

idx (%d)

Job array index.

notifyId (%d)

Identifier or handle for notification.

numResizeHosts (%d)

Number of processors used for execution. If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of hosts listed in short format.

resizeHosts (%s)

List of execution host names. If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format.

numResizeSlots (%d)

Number of allocated slots for executing resize.

resizeSlots (%s)

List of execution host names where slots are allocated for resizing.

JOB_RESIZE_NOTIFY_ACCEPT

LSF logs this event when a resize request has been accepted from the first execution host of a job. The fields in order of occurrence are:

Version number (%s)

The version number.

Event time (%d)

The time of the event.

jobId (%d)

The job ID.

idx (%d)

Job array index.

notifyId (%d)

Identifier or handle for notification.

resizeNotifyCmdPid (%d)

Resize notification executable process ID. If no resize notification executable is defined, this field will be set to 0.

resizeNotifyCmdPGid (%d)

Resize notification executable process group ID. If no resize notification executable is defined, this field will be set to 0.

status (%d)

Status field used to indicate possible errors. 0 Success, 1 failure.

JOB_RESIZE_NOTIFY_DONE

LSF logs this event when the resize notification command completes. The fields in order of occurrence are:

Version number (%s)

The version number.

Event time (%d)

The time of the event.

jobId (%d)

The job ID.

idx (%d)

Job array index.

notifyId (%d)

Identifier or handle for notification.

status (%d)

Resize notification exit value. (0, success, 1, failure, 2 failure but cancel request.)

JOB_RESIZE_RELEASE

LSF logs this event when receiving resource release request from client. The fields in order of occurrence are:

Version number (%s)

The version number.

Event time (%d)

The time of the event.

jobId (%d)

The job ID.

idx (%d)

Job array index.

reqid (%d)

Request Identifier or handle.

options (%d)

Release options.

userId (%d)

UNIX user ID of the user invoking the command.

userName (%s)

User name of the submitter.

resizeNotifyCmd (%s)

Resize notification command to run on the first execution host to inform job of a resize event.

numResizeHosts (%d)

Number of processors used for execution during resize. If

LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of hosts listed in short format.

resizeHosts (%s)

List of execution host names during resize. If **LSF_HPC_EXTENSIONS="SHORT_EVENTFILE"** is specified in `lsf.conf`, the value of this field is logged in a shortened format.

numResizeSlots (%d)

Number of allocated slots for executing resize.

resizeSlots (%s)

List of execution host names where slots are allocated for resizing.

JOB_RESIZE_CANCEL

LSF logs this event when receiving cancel request from client. The fields in order of occurrence are:

Version number (%s)

The version number.

Event time (%d)

The time of the event.

jobId (%d)

The job ID.

idx (%d)

Job array index.

userId (%d)

UNIX user ID of the user invoking the command.

userName (%s)

User name of the submitter.

HOST_POWER_STATUS

LSF logs this event when a host power status is changed, whether by power policy, job, or by the command **badmin hpower**. The **HOST_POWER_STATUS** event is logged to reflect the power status changes. The fields in order of occurrence are:

Version number (%s)

The version number.

Event time (%d)

The time of the event.

Request Id (%d)

The power operation request ID to identify a power operation.

Op Code (%d)

Power operation type.

Trigger (%d)

The power operation trigger: power policy, job, or **badmin hpower**.

Status (%d)

The power operation status.

Trigger Name (%s)

If the operation is triggered by power policy, this is the power policy name. If the operation is triggered by an administrator, this is the administrator user name.

Number (%d)

Number of hosts on which the power operation occurred.

Hosts (%s)

The hosts on which the power operation occurred.

JOB_PROV_HOST

When a job has been dispatched to a power saved host (or hosts), it will trigger a power state change for the host and the job will be in the PROV state. This event logs those PROV cases. The fields in order of occurrence are:

Version number (%s)

The version number.

Event time (%d)

The time of the event.

jobId (%d)

The job ID.

idx (%d)

Job array index.

status (%d)

Indicates if the provision has started, is done, or is failed.

num (%d)

Number of hosts that need to be provisioned.

hostNameList(%d)

Names of hosts that need to be provisioned.

hostStatusList(%d)

Host status for provisioning result.

lsb.globalpolicies

This configuration file defines global policies for multiple clusters.

This file is optional. It is installed by default in `LSB_CONFDIR/cluster_name/configdir`.

GlobalFairshare section

Defines global fairshare policies for multiple clusters.

Each global fairshare policy definition begins with the line `Begin GlobalFairshare` and ends with the line `End GlobalFairshare`. The policy name (**NAME**) and participants (**PARTICIPANTS**) must be specified. All other parameters are optional.

After making any changes to `lsb.globalpolicies`, run **badadmin gpdrestart** to reconfigure the global policy daemon.

The following is the template for this configuration file:

```
# Begin GlobalFairshare
# NAME = policy1
# PARTICIPANTS = queue1@cluster1 queue1@cluster2 queue1@cluster3
# SYNC_MODE = all
# SYNC_INTERVAL = 30
```

```
# DESCRIPTION = A global fairshare policy
# End GlobalFairshare
```

Each line is described in the following subsections.

DESCRIPTION

Syntax

DESCRIPTION=text

Description

Description of the global fairshare policy.

The description should clearly describe the service features of the global fairshare.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\). The maximum length for the text is 512 characters.

Default

Not defined.

NAME

Syntax

NAME=string

Description

Mandatory. Provide a unique name for the global fairshare policy.

Specify any ASCII string up to 60 characters long. You can use letters, digits, underscores (_), dashes (-) and periods (.) in the name.

all is reserved as a keyword.

Default

You must specify this parameter to define a global fairshare policy. LSF does not automatically assign a default global fairshare policy name.

PARTICIPANTS

Syntax

PARTICIPANTS=queue@cluster ...

Description

Mandatory. Define queues for the clusters that participate in the global fairshare policy.

Only the master queue can be configured for user-based cross queue fairshare. The queue name and the cluster name must not include the @ character. A cluster can only have one queue. If **PARTICIPANTS** is "q1@cluster1 q2@cluster1", q2@cluster1 will be ignored.

A participant can only participate in one global fairshare policy. If the participant is already participating in another global fairshare policy, it is ignored in the current global fairshare policy.

Default

Not defined.

SYNC_INTERVAL

Syntax

SYNC_INTERVAL=seconds

Description

Time interval, in seconds, for which the **gpolicyd** and **mbatchd** exchange fairshare load information.

The time interval ranges from 5 seconds to 3600 seconds. If the specified value is out of range, the default value is used.

Default

30 seconds.

SYNC_MODE

Syntax

BSWITCH_SYNC_MODE=all|partial

Description

Defines which share accounts in the fairshare tree of each participant will participate into the global fairshare policy.

- all: If defined, all participants synchronize fairshare load for the whole fairshare tree.
- partial: If defined, depends on the configuration of the **FS_POLICY** setting in `lsb.users` for each cluster. Not all user share accounts in the fairshare tree will synchronize fairshare load globally. Only the defined fairshare node needs global synchronization.

Default

all

lsb.hosts

The `lsb.hosts` file contains host-related configuration information for the server hosts in the cluster. It is also used to define host groups, host partitions, and compute units.

This file is optional. All sections are optional.

By default, this file is installed in `LSB_CONFDIR/cluster_name/configdir`.

Changing lsb.hosts configuration

After making any changes to `lsb.hosts`, run **badmin reconfig** to reconfigure **mbatchd**.

#INCLUDE

Syntax

```
#INCLUDE "path-to-file"
```

Description

Inserts a configuration setting from another file to the current location. Use this directive to dedicate control of a portion of the configuration to other users or user groups by providing write access for the

included file to specific users or user groups, and to ensure consistency of configuration file settings in different clusters (if you are using the LSF multicluster capability).

See more information on shared configuration file content in *Administering IBM Spectrum LSF*

#INCLUDE can be inserted anywhere in the local configuration file.

Default

Not defined.

Host section

Description

Optional. Defines the hosts, host types, and host models used as server hosts, and contains per-host configuration information. If this section is not configured, LSF uses all hosts in the cluster (the hosts listed in `lsf.cluster.cluster_name`) as server hosts.

Each host, host model or host type can be configured to do the following:

- Limit the maximum number of jobs run in total
- Limit the maximum number of jobs run by each user
- Run jobs only under specific load conditions
- Run jobs only under specific time windows

The entries in a line for a host override the entries in a line for its model or type.

When you modify the cluster by adding or removing hosts, no changes are made to `lsb.hosts`. This does not affect the default configuration, but if hosts, host models, or host types are specified in this file, you should check this file whenever you make changes to the cluster and update it manually if necessary.

Host section structure

The first line consists of keywords identifying the load indices that you wish to configure on a per-host basis. The keyword `HOST_NAME` must be used; the others are optional. Load indices not listed on the keyword line do not affect scheduling decisions.

Each subsequent line describes the configuration information for one host, host model or host type. Each line must contain one entry for each keyword. Use empty parentheses () or a dash (-) to specify the default value for an entry.

HOST_NAME

Required. Specify the name, model, or type of a host, or the keyword default.

Pattern definition

You can use string literals and special characters when defining host names. Each entry cannot contain any spaces, as the list itself is space delimited.

You can use the following special characters to specify hosts:

- Use square brackets with a hyphen (`[integer1-integer2]`) or a colon (`[integer1:integer2]`) to define a range of non-negative integers at the end of a host name. The first integer must be less than the second integer.
- Use square brackets with commas (`[integer1, integer2 ...]`) to define individual non-negative integers anywhere in the host name.
- Use square brackets with commas and hyphens or colons (for example, `[integer1-integer2, integer3, integer4:integer5, integer6:integer7]`) to define different ranges of non-negative integers anywhere in the host name.

lsb.hosts

- Use multiple sets of square brackets (with the supported special characters) to define multiple sets of non-negative integers anywhere in the host name. For example, `hostA[1,3]B[1-3]` includes `hostA1B1`, `hostA1B2`, `hostA1B3`, `hostA3B1`, `hostA3B2`, and `hostA3B3`.

host name

The name of a host defined in `lsf.cluster.cluster_name`.

host model

A host model defined in `lsf.shared`.

host type

A host type defined in `lsf.shared`.

default

The reserved host name default indicates all hosts in the cluster not otherwise referenced in the section (by name or by listing its model or type).

CHKPNT

Description

If C, checkpoint copy is enabled. With checkpoint copy, all opened files are automatically copied to the checkpoint directory by the operating system when a process is checkpointed.

Example

```
HOST_NAME  CHKPNT  hostA      C
```

Compatibility

Checkpoint copy is only supported on Cray systems.

Default

No checkpoint copy

DISPATCH_WINDOW

Description

The time windows in which jobs from this host, host model, or host type are dispatched. Once dispatched, jobs are no longer affected by the dispatch window.

Default

Not defined (always open)

EXIT_RATE

Description

Specifies a threshold for exited jobs. Specify a number of jobs. If the number of jobs that exit over a period of time specified by `JOB_EXIT_RATE_DURATION` in `lsb.params` (5 minutes by default) exceeds the number of jobs you specify as the threshold in this parameter, LSF invokes `LSF_SERVERDIR/eadmin` to trigger a host exception.

`EXIT_RATE` for a specific host overrides a default `GLOBAL_EXIT_RATE` specified in `lsb.params`.

Example

The following Host section defines a job exit rate of 20 jobs for all hosts, and an exit rate of 10 jobs on hostA.

```
Begin Host
HOST_NAME    MXJ      EXIT_RATE  # Keywords
Default      !         20
hostA        !         10
End Host
```

Default

Not defined

JL/U**Description**

Per-user job slot limit for the host. Maximum number of job slots that each user can use on this host.

Example

```
HOST_NAME  JL/U
hostA      2
```

Default

Unlimited

MIG**Syntax**

MIG=*minutes*

Description

Enables automatic job migration and specifies the migration threshold for checkpointable or rerunnable jobs, in minutes.

LSF automatically migrates jobs that have been in the SSUSP state for more than the specified number of minutes. Specify a value of 0 to migrate jobs immediately upon suspension. The migration threshold applies to all jobs running on the host.

Job-level command line migration threshold overrides threshold configuration in application profile and queue. Application profile configuration overrides queue level configuration. When a host migration threshold is specified, and is lower than the value for the job, the queue, or the application, the host value is used.

Does not affect MultiCluster jobs that are forwarded to a remote cluster.

Default

Not defined. LSF does not migrate checkpointable or rerunnable jobs automatically.

MXJ**Description**

The number of job slots on the host.

With MultiCluster resource leasing model, this is the number of job slots on the host that are available to the local cluster.

lsb.hosts

Use ! to make the number of job slots equal to the number of CPUs on a host.

For the reserved host name default, ! makes the number of job slots equal to the number of CPUs on all hosts in the cluster not otherwise referenced in the section.

By default, the number of running and suspended jobs on a host cannot exceed the number of job slots. If preemptive scheduling is used, the suspended jobs are not counted as using a job slot.

On multiprocessor hosts, to fully use the CPU resource, make the number of job slots equal to or greater than the number of processors.

Default

Unlimited

load_index

Syntax

```
load_index loadSched[/loadStop]
```

Specify io, it, ls, mem, pg, r15s, r1m, r15m, swp, tmp, ut, or a non-shared (host based) dynamic custom external load index as a column. Specify multiple columns to configure thresholds for multiple load indices.

Description

Scheduling and suspending thresholds for dynamic load indices supported by LIM, including external load indices.

Each load index column must contain either the default entry or two numbers separated by a slash (/), with no white space. The first number is the scheduling threshold for the load index; the second number is the suspending threshold.

Queue-level scheduling and suspending thresholds are defined in lsb.queues. If both files specify thresholds for an index, those that apply are the most restrictive ones.

Example

```
HOST_NAME    mem      swp
hostA        100/10  200/30
```

This example translates into a loadSched condition of

```
mem>=100 && swp>=200
```

and a loadStop condition of

```
mem < 10 || swp < 30
```

Default

Not defined

AFFINITY

Syntax

```
AFFINITY=Y|y|N|n|cpu_list
```

Description

Specifies whether the host can be used to run affinity jobs, and if so which CPUs are eligible to do so. The syntax accepts Y, N, a list of CPUs, or a CPU range.

Examples

The following configuration enables affinity scheduling and tells LSF to use all CPUs on hostA for affinity jobs:

```
HOST_NAME MXJ r1m AFFINITY
hostA      !  ( )  (Y)
```

The following configuration specifies a CPU list for affinity scheduling:

```
HOST_NAME MXJ r1m  AFFINITY
hostA      !  ( )  (CPU_LIST="1,3,5,7-10")
```

This configuration enables affinity scheduling on hostA and tells LSF to just use CPUs 1,3,5, and CPUs 7-10 to run affinity jobs.

The following configuration disables affinity scheduling:

```
HOST_NAME MXJ r1m AFFINITY
hostA      !  ( )  (N)
```

Default

Not defined. Affinity scheduling is not enabled.

Example of a Host section

```
Begin Host
HOST_NAME  MXJ  JL/U  r1m      pg      DISPATCH_WINDOW
hostA      1    -    0.6/1.6  10/20  (5:19:00-1:8:30 20:00-8:30)
Linux     1    -    0.5/2.5  -      23:00-8:00
default   2    1    0.6/1.6  20/40  ( )
End Host
```

Linux is a host type defined in `lsf.shared`. This example Host section configures one host and one host type explicitly and configures default values for all other load-sharing hosts.

HostA runs one batch job at a time. A job will only be started on hostA if the `r1m` index is below 0.6 and the `pg` index is below 10; the running job is stopped if the `r1m` index goes above 1.6 or the `pg` index goes above 20. HostA only accepts batch jobs from 19:00 on Friday evening until 8:30 Monday morning and overnight from 20:00 to 8:30 on all other days.

For hosts of type Linux, the `pg` index does not have host-specific thresholds and such hosts are only available overnight from 23:00 to 8:00.

The entry with host name default applies to each of the other hosts in the cluster. Each host can run up to two jobs at the same time, with at most one job from each user. These hosts are available to run jobs at all times. Jobs may be started if the `r1m` index is below 0.6 and the `pg` index is below 20.

HostGroup section

Description

Optional. Defines host groups.

The name of the host group can then be used in other host group, host partition, and queue definitions, as well as on the command line. Specifying the name of a host group has exactly the same effect as listing the names of all the hosts in the group.

Structure

Host groups are specified in the same format as user groups in `lsb.users`.

The first line consists of two mandatory keywords, `GROUP_NAME` and `GROUP_MEMBER`, as well as optional keywords, `CONDENSE` and `GROUP_ADMIN`. Subsequent lines name a group and list its membership.

The sum of all host groups, compute groups, and host partitions cannot be more than 1024.

GROUP_NAME

Description

An alphanumeric string representing the name of the host group.

You cannot use the reserved name `all`, and group names must not conflict with host names.

CONDENSE

Description

Optional. Defines condensed host groups.

Condensed host groups are displayed in a condensed output format for the **bhosts** and **bjobs** commands.

If you configure a host to belong to more than one condensed host group, **bjobs** can display any of the host groups as execution host name.

Valid values

Y or N.

Default

N (the specified host group is not condensed)

GROUP_MEMBER

Description

A space-delimited list of host names or previously defined host group names, enclosed in one pair of parentheses.

You cannot use more than one pair of parentheses to define the list.

The names of hosts and host groups can appear on multiple lines because hosts can belong to multiple groups. The reserved name `all` specifies all hosts in the cluster. An exclamation mark (!) indicates an externally-defined host group, which the **egroup** executable retrieves.

Pattern definition

You can use string literals and special characters when defining host group members. Each entry cannot contain any spaces, as the list itself is space delimited.

When a leased-in host joins the cluster, the host name is in the form of *host@cluster*. For these hosts, only the host part of the host name is subject to pattern definitions.

You can use the following special characters to specify host group members:

- Use a tilde (~) to exclude specified hosts or host groups from the list.
- Use an asterisk (*) as a wildcard character to represent any number of characters.

- Use square brackets with a hyphen (*[integer1-integer2]*) or a colon (*[integer1:integer2]*) to define a range of non-negative integers anywhere in the host name. The first integer must be less than the second integer.
- Use square brackets with commas (*[integer1, integer2 ...]*) to define individual non-negative integers anywhere in the host name.
- Use square brackets with commas and hyphens or colons (for example, *[integer1-integer2, integer3, integer4:integer5, integer6:integer7]*) to define different ranges of non-negative integers anywhere in the host name.
- Use multiple sets of square brackets (with the supported special characters) to define multiple sets of non-negative integers anywhere in the host name. For example, *hostA[1,3]B[1-3]* includes *hostA1B1*, *hostA1B2*, *hostA1B3*, *hostA3B1*, *hostA3B2*, and *hostA3B3*.

Restrictions

You cannot define subgroups that contain wildcards and special characters.

GROUP_ADMIN

Description

Host group administrators have the ability to open or close the member hosts for the group they are administering.

the `GROUP_ADMIN` field is a space-delimited list of user names or previously defined user group names, enclosed in one pair of parentheses.

You cannot use more than one pair of parentheses to define the list.

The names of users and user groups can appear on multiple lines because users can belong to and administer multiple groups.

Host group administrator rights are inherited. For example, if the user `admin2` is an administrator for host group `hg1` and host group `hg2` is a member of `hg1`, `admin2` is also an administrator for host group `hg2`.

When host group administrators (who are not also cluster administrators) open or close a host, they must specify a comment with the `-C` option.

Valid values

Any existing user or user group can be specified. A user group that specifies an external list is also allowed; however, in this location, you use the user group name that has been defined with `(!)` rather than `(!)` itself.

Restrictions

- You cannot specify any wildcards or special characters (for example: `*`, `!`, `$`, `#`, `&`, `~`).
- You cannot specify an external group (egroup).
- You cannot use the keyword `ALL` and you cannot administer any group that has `ALL` as its members.
- User names and user group names cannot have spaces.

Example HostGroup sections

Example 1

```
Begin HostGroup
GROUP_NAME  GROUP_MEMBER GROUP_ADMIN
groupA      (hostA hostD) (user1 user10)
groupB      (hostF groupA hostK) ()
groupC      (!) ()
End HostGroup
```

This example defines three host groups:

- `groupA` includes `hostA` and `hostD` and can be administered by `user1` and `user10`.
- `groupB` includes `hostF` and `hostK`, along with all hosts in `groupA`. It has no administrators (only the cluster administrator can control the member hosts).
- The group membership of `groupC` is defined externally and retrieved by the **egroup** executable.

Example 2

```
Begin HostGroup
GROUP_NAME  GROUP_MEMBER GROUP_ADMIN
groupA      (all) ()
groupB      (groupA ~hostA ~hostB) (user11 user14)
groupC      (hostX hostY hostZ) ()
groupD      (groupC ~hostX) usergroupB
groupE      (all ~groupC ~hostB) ()
groupF      (hostF groupC hostK) ()
End HostGroup
```

This example defines the following host groups:

- `groupA` contains all hosts in the cluster and is administered by the cluster administrator.
- `groupB` contains all the hosts in the cluster except for `hostA` and `hostB` and is administered by `user11` and `user14`.
- `groupC` contains only `hostX`, `hostY`, and `hostZ` and is administered by the cluster administrator.
- `groupD` contains the hosts in `groupC` except for `hostX`. Note that `hostX` must be a member of host group `groupC` to be excluded from `groupD`. `usergroupB` is the administrator for `groupD`.
- `groupE` contains all hosts in the cluster excluding the hosts in `groupC` and `hostB` and is administered by the cluster administrator.
- `groupF` contains `hostF`, `hostK`, and the 3 hosts in `groupC` and is administered by the cluster administrator.

Example 3

```
Begin HostGroup
GROUP_NAME  CONDENSE  GROUP_MEMBER GROUP_ADMIN
groupA      N          (all) ()
groupB      N          (hostA, hostB) (usergroupC user1)
groupC      Y          (all)()
End HostGroup
```

This example defines the following host groups:

- `groupA` shows uncondensed output and contains all hosts in the cluster and is administered by the cluster administrator.
- `groupB` shows uncondensed output, and contains `hostA` and `hostB`. It is administered by all members of `usergroupC` and `user1`.
- `groupC` shows condensed output and contains all hosts in the cluster and is administered by the cluster administrator.

Example 4

```
Begin HostGroup
GROUP_NAME CONDENSE GROUP_MEMBER GROUP_ADMIN
groupA      Y (host*) (user7)
groupB      N (*A) ()
groupC      N (hostB* ~hostB[1-50]) ()
groupD      Y (hostC[1:50] hostC[101:150]) (usergroupJ)
groupE      N (hostC[51-100] hostC[151-200]) ()
groupF      Y (hostD[1,3] hostD[5-10]) ()
groupG      N (hostD[11-50] ~hostD[15,20,25] hostD2) ()
```

```
groupH          Y (hostX[1:10]Y[1:10]) ()
End HostGroup
```

This example defines the following host groups:

- `groupA` shows condensed output, and contains all hosts starting with the string `host`. It is administered by `user7`.
- `groupB` shows uncondensed output, and contains all hosts ending with the string `A`, such as `hostA` and is administered by the cluster administrator.
- `groupC` shows uncondensed output, and contains all hosts starting with the string `hostB` except for the hosts from `hostB1` to `hostB50` and is administered by the cluster administrator.
- `groupD` shows condensed output, and contains all hosts from `hostC1` to `hostC50` and all hosts from `hostC101` to `hostC150` and is administered by the the members of `usergroupJ`.
- `groupE` shows uncondensed output, and contains all hosts from `hostC51` to `hostC100` and all hosts from `hostC151` to `hostC200` and is administered by the cluster administrator.
- `groupF` shows condensed output, and contains `hostD1`, `hostD3`, and all hosts from `hostD5` to `hostD10` and is administered by the cluster administrator.
- `groupG` shows uncondensed output, and contains all hosts from `hostD11` to `hostD50` except for `hostD15`, `hostD20`, and `hostD25`. `groupG` also includes `hostD2`. It is administered by the cluster administrator.

HostPartition section

Description

Optional. Used with host partition user-based fairshare scheduling. Defines a host partition, which defines a user-based fairshare policy at the host level.

Configure multiple sections to define multiple partitions.

The members of a host partition form a host group with the same name as the host partition.

Restriction: You cannot use host partitions and host preference simultaneously.

Limitations on queue configuration

- If you configure a host partition, you cannot configure fairshare at the queue level.
- If a queue uses a host that belongs to a host partition, it should not use any hosts that don't belong to that partition. All the hosts in the queue should belong to the same partition. Otherwise, you might notice unpredictable scheduling behavior:
 - Jobs in the queue sometimes may be dispatched to the host partition even though hosts not belonging to any host partition have a lighter load.
 - If some hosts belong to one host partition and some hosts belong to another, only the priorities of one host partition are used when dispatching a parallel job to hosts from more than one host partition.

Shared resources and host partitions

- If a resource is shared among hosts included in host partitions and hosts that are not included in any host partition, jobs in queues that use the host partitions will always get the shared resource first, regardless of queue priority.
- If a resource is shared among host partitions, jobs in queues that use the host partitions listed first in the `HostPartition` section of `lsb.hosts` will always have priority to get the shared resource first. To allocate shared resources among host partitions, LSF considers host partitions in the order they are listed in `lsb.hosts`.

Structure

Each host partition always consists of 3 lines, defining the name of the partition, the hosts included in the partition, and the user share assignments.

HPART_NAME**Syntax**

HPART_NAME=*partition_name*

Description

Specifies the name of the partition. The name must be 59 characters or less.

HOSTS**Syntax**

HOSTS=[[~]*host_name* / [~]*host_group* / all]...

Description

Specifies the hosts in the partition, in a space-separated list.

A host cannot belong to multiple partitions.

A host group cannot be empty.

Hosts that are not included in any host partition are controlled by the FCFS scheduling policy instead of the fairshare scheduling policy.

Optionally, use the reserved host name all to configure a single partition that applies to all hosts in a cluster.

Optionally, use the not operator (~) to exclude hosts or host groups from the list of hosts in the host partition.

Examples

```
HOSTS=all ~hostK ~hostM
```

The partition includes all the hosts in the cluster, except for hostK and hostM.

```
HOSTS=groupA ~hostL
```

The partition includes all the hosts in host group groupA except for hostL.

USER_SHARES**Syntax**

USER_SHARES=[*user*, *number_shares*]...

Description

Specifies user share assignments

- Specify at least one user share assignment.
- Enclose each user share assignment in square brackets, as shown.
- Separate a list of multiple share assignments with a space between the square brackets.
- *user*—Specify users who are also configured to use the host partition. You can assign the shares:

- To a single user (specify *user_name*). To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).
- To users in a group, individually (specify *group_name@*) or collectively (specify *group_name*). To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME\group_name*).
- To users not included in any other share assignment, individually (specify the keyword *default*) or collectively (specify the keyword *others*).

By default, when resources are assigned collectively to a group, the group members compete for the resources according to FCFS scheduling. You can use hierarchical fairshare to further divide the shares among the group members.

When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.

- *number_shares*

- Specify a positive integer representing the number of shares of the cluster resources assigned to the user.
- The number of shares assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares assigned in each share assignment.

Example of a HostPartition section

```
Begin HostPartition
HPART_NAME = Partition1 HOSTS = hostA hostB USER_SHARES =
[groupA@, 3] [groupB, 7] [default, 1]
End HostPartition
```

ComputeUnit section

Description

Optional. Defines compute units.

Once defined, the compute unit can be used in other compute unit and queue definitions, as well as in the command line. Specifying the name of a compute unit has the same effect as listing the names of all the hosts in the compute unit.

Compute units are similar to host groups, with the added feature of granularity allowing the construction of structures that mimic the network architecture. Job scheduling using compute unit resource requirements effectively spreads jobs over the cluster based on the configured compute units.

To enforce consistency, compute unit configuration has the following requirements:

- Hosts and host groups appear in the finest granularity compute unit type, and nowhere else.
- Hosts appear in only one compute unit of the finest granularity.
- All compute units of the same type have the same type of compute units (or hosts) as members.

Structure

Compute units are specified in the same format as host groups in `lsb.hosts`.

The first line consists of three mandatory keywords, `NAME`, `MEMBER`, and `TYPE`, as well as optional keywords `CONDENSE` and `ADMIN`. Subsequent lines name a compute unit and list its membership.

The sum of all host groups, compute groups, and host partitions cannot be more than 1024.

NAME**Description**

An alphanumeric string representing the name of the compute unit.

You cannot use the reserved names `all`, `allremote`, `others`, and `default`. Compute unit names must not conflict with host names, host partitions, or host group names.

CONDENSE**Description**

Optional. Defines condensed compute units.

Condensed compute units are displayed in a condensed output format for the **bhosts** and **bjobs** commands. The condensed compute unit format includes the slot usage for each compute unit.

Valid values

Y or N.

Default

N (the specified host group is not condensed)

MEMBER**Description**

A space-delimited list of host names or previously defined compute unit names, enclosed in one pair of parentheses.

You cannot use more than one pair of parentheses to define the list.

The names of hosts and host groups can appear only once, and only in a compute unit type of the finest granularity.

An exclamation mark (!) indicates an externally-defined host group, which the **egroup** executable retrieves.

Pattern definition

You can use string literals and special characters when defining compute unit members. Each entry cannot contain any spaces, as the list itself is space delimited.

You can use the following special characters to specify host and host group compute unit members:

- Use a tilde (~) to exclude specified hosts or host groups from the list.
- Use an asterisk (*) as a wildcard character to represent any number of characters.
- Use square brackets with a hyphen (*[integer1-integer2]*) or a colon (*[integer1:integer2]*) to define a range of non-negative integers anywhere in the host name. The first integer must be less than the second integer.
- Use square brackets with commas (*[integer1, integer2...]*) to define individual non-negative integers anywhere in the host name.
- Use square brackets with commas and hyphens or colons (for example, *[integer1-integer2, integer3, integer4:integer5, integer6:integer7]*) to define different ranges of non-negative integers anywhere in the host name.
- Use multiple sets of square brackets (with the supported special characters) to define multiple sets of non-negative integers anywhere in the host name. For example, `hostA[1,3]B[1-3]` includes `hostA1B1`, `hostA1B2`, `hostA1B3`, `hostA3B1`, `hostA3B2`, and `hostA3B3`.

Restrictions

- Compute unit names cannot be used in compute units of the finest granularity.
- You cannot include host or host group names except in compute units of the finest granularity.
- You must not skip levels of granularity. For example:
If `lsb.params` contains `COMPUTE_UNIT_TYPES=enclosure rack cabinet` then a compute unit of type `cabinet` can contain compute units of type `rack`, but not of type `enclosure`.
- The keywords `all`, `allremote`, `all@cluster`, `other` and `default` cannot be used when defining compute units.

TYPE**Description**

The type of the compute unit, as defined in the **COMPUTE_UNIT_TYPES** parameter of `lsb.params`.

ADMIN**Description**

Compute unit administrators have the ability to open or close the member hosts for the compute unit they are administering.

the `ADMIN` field is a space-delimited list of user names or previously defined user group names, enclosed in one pair of parentheses.

You cannot use more than one pair of parentheses to define the list.

The names of users and user groups can appear on multiple lines because users can belong to and administer multiple compute units.

Compute unit administrator rights are inherited. For example, if the user `admin2` is an administrator for compute unit `cu1` and compute unit `cu2` is a member of `cu1`, `admin2` is also an administrator for compute unit `cu2`.

When compute unit administrators (who are not also cluster administrators) open or close a host, they must specify a comment with the `-C` option.

Valid values

Any existing user or user group can be specified. A user group that specifies an external list is also allowed; however, in this location, you use the user group name that has been defined with `(!)` rather than `(!)` itself.

Restrictions

- You cannot specify any wildcards or special characters (for example: `*`, `!`, `$`, `#`, `&`, `~`).
- You cannot specify an external group (egroup).
- You cannot use the keyword `ALL` and you cannot administer any group that has `ALL` as its members.
- User names and user group names cannot have spaces.

Example ComputeUnit sections**Example 1**

(For the `lsb.params` entry `COMPUTE_UNIT_TYPES=enclosure rack cabinet`)

```
Begin ComputeUnit
NAME  MEMBER      TYPE
enc11 (host1 host2) enclosure
enc12 (host3 host4) enclosure
```

```

encl3 (host5 host6) enclosure
encl4 (host7 host8) enclosure
rack1 (encl1 encl2) rack
rack2 (encl3 encl4) rack
cbnt1 (rack1 rack2) cabinet
End ComputeUnit

```

This example defines seven compute units:

- encl1, encl2, encl3 and encl4 are the finest granularity, and each contain two hosts.
- rack1 is of coarser granularity and contains two levels. At the enclosure level rack1 contains encl1 and encl2. At the lowest level rack1 contains host1, host2, host3, and host4.
- rack2 has the same structure as rack1, and contains encl3 and encl4.
- cbnt1 contains two racks (rack1 and rack2), four enclosures (encl1, encl2, encl3, and encl4) and all eight hosts. Compute unit cbnt1 is the coarsest granularity in this example.

Example 2

(For the lsb.params entry COMPUTE_UNIT_TYPES=enclosure rack cabinet)

```

Begin ComputeUnit
NAME CONDENSE MEMBER TYPE ADMIN
encl1 Y (hg123 ~hostA ~hostB) enclosure (user11 user14)
encl2 Y (hg456) enclosure ()
encl3 N (hostA hostB) enclosure usergroupB
encl4 N (hgroupX ~hostB) enclosure ()
encl5 Y (hostC* ~hostC[101-150]) enclosure usergroupJ
encl6 N (hostC[101-150]) enclosure ()
rack1 Y (encl1 encl2 encl3) rack ()
rack2 N (encl4 encl5) rack usergroupJ
rack3 N (encl6) rack ()
cbnt1 Y (rack1 rack2) cabinet ()
cbnt2 N (rack3) cabinet user14
End ComputeUnit

```

This example defines 11 compute units:

- All six enclosures (finest granularity) contain only hosts and host groups. All three racks contain only enclosures. Both cabinets (coarsest granularity) contain only racks.
- encl1 contains all the hosts in host group hg123 except for hostA and hostB and is administered by user11 and user14. Note that hostA and hostB must be members of host group hg123 to be excluded from encl1. encl1 shows condensed output.
- encl2 contains host group hg456 and is administered by the cluster administrator. encl2 shows condensed output.
- encl3 contains hostA and hostB. usergroupB is the administrator for encl3. encl3 shows uncondensed output.
- encl4 contains host group hgroupX except for hostB. Since each host can appear in only one enclosure and hostB is already in encl3, it cannot be in encl4. encl4 is administered by the cluster administrator. encl4 shows uncondensed output.
- encl5 contains all hosts starting with the string hostC except for hosts hostC101 to hostC150, and is administered by usergroupJ. encl5 shows condensed output.
- rack1 contains encl1, encl2, and encl3. rack1 shows condensed output.
- rack2 contains encl4, and encl5. rack2 shows uncondensed output.
- rack3 contains encl6. rack3 shows uncondensed output.
- cbnt1 contains rack1 and rack2. cbnt1 shows condensed output.
- cbnt2 contains rack3. Even though rack3 only contains encl6, cbnt3 cannot contain encl6 directly because that would mean skipping the level associated with compute unit type rack. cbnt2 shows uncondensed output.

Automatic time-based configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in `lsb.hosts` by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the **admin reconfig** command.

The expressions are evaluated by LSF every 10 minutes based on `mbatchd` start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting `mbatchd`, providing continuous system availability.

Example

In the following example, the `#if`, `#else`, `#endif` are not interpreted as comments by LSF but as if-else constructs.

```
Begin Host
HOST_NAME    r15s    r1m    pg
host1        3/5     3/5     12/20
#if time(5:16:30-1:8:30 EDT 20:00-8:30 EDT)
host2        3/5     3/5     12/20
#else
host2        2/3     2/3     10/12
#endif
host3        3/5     3/5     12/20
End Host
```

Specifying the time zone is optional. If you do not specify a time zone, LSF uses the local system time zone. LSF supports all standard time zone abbreviations.

lsb.modules

The `lsb.modules` file contains configuration information for LSF scheduler and resource broker modules. The file contains only one section, named `PluginModule`.

This file is optional. If no scheduler or resource broker modules are configured, LSF uses the default scheduler plug-in modules named `schmod_default` and `schmod_fcfs`.

The `lsb.modules` file is stored in the directory `LSB_CONFDIR/cluster_name/configdir`. The **LSB_CONFDIR** parameter is defined in the `lsf.conf` file.

After you change lsb.modules configuration

After you change the `lsb.modules` file, run the **admin reconfig** command to reconfigure the `mbatchd` daemon.

PluginModule section

Description

The `PluginModule` section defines the plug-in modules for the LSF scheduler and LSF resource broker. If the `PluginModule` section is not configured, LSF uses the default scheduler plug-in modules that are named `schmod_default` and `schmod_fcfs`, which enable the LSF default scheduling features.

Example PluginModule section

The following `PluginModule` section enables all scheduling policies provided by LSF:

```
Begin PluginModule
SCH_PLUGIN    RB_PLUGIN    SCH_DISABLE_PHASES
schmod_default    ()            ()
schmod_fairshare  ()            ()
schmod_fcfs       ()            ()
schmod_limit      ()            ()
schmod_parallel   ()            ()
schmod_reserve    ()            ()
```

```

schmod_preemption      ()
schmod_advisv          ()
schmod_mc              ()
schmod_jobweight      ()
schmod_cpuset         ()
schmod_pset           ()
schmod_ps             ()
schmod_aps            ()
schmod_affinity       ()
schmod_demand         ()
End PluginModule

```

PluginModule section structure

The first line consists of the following keywords:

- SCH_PLUGIN
- RB_PLUGIN
- SCH_DISABLE_PHASES

They identify the scheduler plug-ins, resource broker plug-ins, and the scheduler phase to be disabled for the plug-ins that you want to configure.

Each line describes the configuration information for one scheduler plug-in module, resource broker plug-in module, and scheduler phase, if any, to be disabled for the plug-in. Each line must contain one entry for each keyword. Use empty parentheses () or a dash (-) to specify the default value for an entry.

SCH_PLUGIN

Description

Required. The SCH_PLUGIN column specifies the shared module name for the LSF scheduler plug-in. Scheduler plug-ins are called in the order they are listed in the PluginModule section.

By default, all shared modules for scheduler plug-ins are located in the directory defined by the **LSF_LIBDIR** parameter in the `lsf.conf` file. On UNIX and Linux, you can also specify a full path to the name of the scheduler plug-in.

The following modules are supplied with LSF:

schmod_default

Enables the default LSF scheduler features.

schmod_fcfs

Enables the first-come, first-served (FCFS) scheduler features. `schmod_fcfs` can appear anywhere in the SCH_PLUGIN list. By default, if `schmod_fcfs` is not configured in the `lsb.modules` file, it is loaded automatically along with `schmod_default`.

Source code (`sch.mod.fcfs.c`) for the `schmod_fcfs` scheduler plug-in module is installed in the directory

```
LSF_TOP/10.1/misc/examples/external_plugin/
```

Use the LSF scheduler plug-in SDK to modify the FCFS scheduler module code to suit the job scheduling requirements of your site.

schmod_fairshare

Enables the LSF fairshare scheduling features.

schmod_limit

Enables the LSF resource allocation limit features.

schmod_parallel

Enables scheduling of parallel jobs submitted with **bsub -n**.

schmod_reserve

Enables the LSF resource reservation features.

To enable processor reservation, backfill, and memory reservation for parallel jobs, you must configure both `schmod_parallel` and `schmod_reserve` in `lsb.modules`. If only `schmod_reserve` is configured, backfill and memory reservation are enabled only for sequential jobs, and processor reservation is not enabled.

schmod_preemption

Enables the LSF preemption scheduler features.

schmod_advrsv

Handles jobs that use advance reservation (**brsvadd, brsvs, brsvdel, bsub -U**)

schmod_cpuset

Handles jobs that use SGI cpusets (**bsub -ext[sched] "CPUSET[cpuset_options]"**)

The `schmod_cpuset` plug-in name must be configured after the standard LSF plug-in names in the `PluginModule` list.

schmod_mc

Enables IBM Spectrum LSF multicluster capability job forwarding

schmod_ps

Enables resource ownership functionality of EGO-enabled SLA scheduling policies

schmod_aps

Enables absolute priority scheduling (APS) policies configured by the **APS_PRIORITY** parameter in the `lsb.queues` file.

The `schmod_aps` plug-in name must be configured after the `schmod_fairshare` plug-in name in the `PluginModule` list so that the APS value can override the fairshare job-ordering decision.

schmod_affinity

Enables configured by the **AFFINITY** parameter in the `lsf.conf` file.

schmod_demand

Enables LSF resource connector functionality configured by the **LSB_RC_EXTERNAL_HOST_FLAG** parameter in the `lsf.conf` file. With LSF resource connector enabled, clusters borrow resources from IBM Spectrum Conductor with Spark based on workload.

Scheduler plug-in SDK

Use the LSF scheduler plug-in SDK to write customized scheduler modules that give you more flexibility and control over job scheduling. Enable your custom scheduling policies by configuring your modules under `SCH_PLUGIN` in the `PluginModules` section of the `lsb.modules` file.

The following directory contains sample plug-in code:

```
LSF_TOP/10.1/misc/examples/external_plugin/
```

SCH_DISABLE_PHASES

Description

The SCH_DISABLE_PHASES column lists the scheduler phases, if any, to be disabled for the plug-in. LSF scheduling has four phases:

1. Preprocessing - the scheduler checks the readiness of the job for scheduling and prepares a list of ready resource seekers. It also checks the start time of a job, and evaluates any job dependencies.
2. Match/limit - the scheduler evaluates the job resource requirements and prepares candidate hosts for jobs by matching jobs with resources. It also applies resource allocation limits. Jobs with all required resources matched go on to order/allocation phase. Not all jobs are mapped to all potential available resources. Jobs without any matching resources will not go through the Order/Allocation Phase but can go through the Post-processing phase, where preemption might be applied to get resources the job needs to run.
3. Order/allocation - the scheduler sorts jobs with matched resources and allocates resources for each job, assigning job slot, memory, and other resources to the job. It also checks whether the allocation satisfies all constraints defined in configuration, such as queue slot limit, deadline for the job.
 - a. In the order phase, the scheduler applies policies such as FCFS, Fairshare, and Host-partition and consider job priorities within user groups and share groups. By default, job priority within a pool of jobs from the same user is based on how long the job has been pending.
 - b. For resource-intensive jobs (jobs that require many CPUs or a large amount of memory), resources are reserved so that these jobs are not starved.
 - c. When all the currently available resources are allocated, jobs go on to post-processing.
4. Post-processing - the scheduler prepares jobs from the order/allocation phase for dispatch. Post-processing also applies preemption or backfill policies to obtain resources for the jobs with pre-processing or match/limit phases that are complete, but which did not have resources available to enter the next scheduling phase.

Each scheduler plug-in module invokes one or more scheduler phases. The processing for a give phase can be disabled or skipped if the plug-in module does not need to do any processing for that phase or the processing is already done by a previous plug-in module in the list.

Default

Undefined

lsb.params

The `lsb.params` file defines general parameters used by the LSF system. This file contains only one section, named Parameters. **mbatchd** uses `lsb.params` for initialization. The file is optional. If not present, the LSF-defined defaults are assumed.

Some of the parameters that can be defined in `lsb.params` control timing within the system. The default settings provide good throughput for long-running batch jobs while adding a minimum of processing overhead in the batch daemons.

This file is installed by default in `LSB_CONFDIR/cluster_name/configdir`.

Changing lsb.params configuration

After making any changes to `lsb.params`, run **badadmin reconfig** to reconfigure `mbatchd`.

Automatic time-based configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in `lsb.params` by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the **admin reconfig** command.

The expressions are evaluated by LSF every 10 minutes based on `mbatchd` start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting `mbatchd`, providing continuous system availability.

Example

```
# if 18:30-19:30 is your short job express period, but
# you want all jobs going to the short queue by default
# and be subject to the thresholds of that queue
# for all other hours, normal is the default queue
#if time(18:30-19:30 EDT)
DEFAULT_QUEUE=short
#else
DEFAULT_QUEUE=normal
#endif
```

Specifying the time zone is optional. If you do not specify a time zone, LSF uses the local system time zone. LSF supports all standard time zone abbreviations.

Parameters section

This section and all the keywords in this section are optional. If keywords are not present, the default values are assumed.

Parameters set at installation

The following parameter values are set at installation for the purpose of testing a new cluster:

```
Begin Parameters
DEFAULT_QUEUE = normal    #default job queue name
MBD_SLEEP_TIME = 10      #Time used for calculating parameter values (60 secs is default)
SBD_SLEEP_TIME = 7       #sbatchd scheduling interval (30 secs is default)
JOB_ACCEPT_INTERVAL = 1  #interval for any host to accept a job
                          #(default is 1 (one-fold of MBD_SLEEP_TIME))
End Parameters
```

With this configuration, jobs submitted to the LSF system will be started on server hosts quickly. If this configuration is not suitable for your production use, you should either remove the parameters to take the default values, or adjust them as needed.

For example, to avoid having jobs start when host load is high, increase **JOB_ACCEPT_INTERVAL** so that the job scheduling interval is longer to give hosts more time to adjust load indices after accepting jobs.

In production use, you should define **DEFAULT_QUEUE** to the normal queue, **MBD_SLEEP_TIME** to 60 seconds (the default), and **SBD_SLEEP_TIME** to 30 seconds (the default).

ABS_RUNLIMIT

Syntax

```
ABS_RUNLIMIT=y | Y
```

Description

If set, absolute (wall-clock) run time is used instead of normalized run time for all jobs submitted with the following values:

- Run time limit or run time estimate specified by the `-W` or `-We` option of **bsub**

lsb.params

- **RUNLIMIT** queue-level parameter in `lsb.queues`
- **RUNLIMIT** application-level parameter in `lsb.applications`
- **RUNTIME** parameter in `lsb.applications`

If the CPU factor for a submission host is changed, then the **RUNLIMIT** for a job that is already submitted will be recalculated.

Note: When **ABS_RUNLIMIT** is enabled, the **bqueues -l** command does not show the run limit normalized to the host.

Default

Set to Y at time of installation. If otherwise undefined, then N.

ACCT_ARCHIVE_AGE

Syntax

ACCT_ARCHIVE_AGE=*days*

Description

Enables automatic archiving of LSF accounting log files, and specifies the archive interval. LSF archives the current log file if the length of time from its creation date exceeds the specified number of days.

See also

- **ACCT_ARCHIVE_SIZE** also enables automatic archiving
- **ACCT_ARCHIVE_TIME** also enables automatic archiving
- **MAX_ACCT_ARCHIVE_FILE** enables automatic deletion of the archives

Default

Not defined; no limit to the age of `lsb.acct`.

ACCT_ARCHIVE_SIZE

Syntax

ACCT_ARCHIVE_SIZE=*kilobytes*

Description

Enables automatic archiving of LSF accounting log files, and specifies the archive threshold. LSF archives the current log file if its size exceeds the specified number of kilobytes.

See also

- **ACCT_ARCHIVE_SIZE** also enables automatic archiving
- **ACCT_ARCHIVE_TIME** also enables automatic archiving
- **MAX_ACCT_ARCHIVE_FILE** enables automatic deletion of the archives

Default

Not defined; no limit to the size of `lsb.acct`.

ACCT_ARCHIVE_TIME

Syntax

ACCT_ARCHIVE_TIME=*hh:mm*

Description

Enables automatic archiving of LSF accounting log file `lsb.acct`, and specifies the time of day to archive the current log file.

See also

- **ACCT_ARCHIVE_SIZE** also enables automatic archiving
- **ACCT_ARCHIVE_TIME** also enables automatic archiving
- **MAX_ACCT_ARCHIVE_FILE** enables automatic deletion of the archives

Default

Not defined (no time set for archiving `lsb.acct`)

ADVRSV_USER_LIMIT

Syntax

ADVRSV_USER_LIMIT=*integer*

Description

Sets the number of advance reservations each user or user group can have in the system.

Valid values

1-10000

Default

100

ALLOCATION_PLANNER

Syntax

ALLOCATION_PLANNER=Y | N | "<key>[value] ..."

Description

By itself, this parameter does not have any effect on scheduling behavior. You must separately identify sets of candidate jobs to consider in the planner. This is done by setting **PLAN=Y** at the application level (`lsb.applications` file), queue level (`lsb.queues` file), or cluster level (`lsb.params` file).

The following key-value pairs are supported:

Table 2. Key-Value pairs for ALLOCATION_PLANNER			
key	value	Default	Description
WINDOW	positive integer	60	Number of minutes the planner looks into the future.

key	value	Default	Description
MAX_TIME	positive integer	3	Number of minute that the planner is permitted to spend searching for new planned allocations.

Note:

The **PLAN** parameter replaces the existing **SLOT_RESERVE** parameter and **RESOURCE_RESERVE** parameter when the **ALLOCATION_PLANNER** parameter is enabled.

Default

N

ALLOW_EVENT_TYPE

Specifies the events that are logged into the stream files (lsb.stream, lsb.status, and lsb.pending reasons).

Syntax

ALLOW_EVENT_TYPE=event_type1 [event_type2 ...]

Description

Use this parameter to prevent unwanted events from being logged into the stream files.

Valid values

Any space-separated list of valid LSF event types.

Default

Not defined. All event types are logged in the stream files.

Example

The following example allows only the **JOB_NEW**, **JOB_STATUS**, and **JOB_FINISH** events to be logged in the stream files:

```
ALLOW_EVENT_TYPE=JOB_NEW JOB_STATUS JOB_FINISH
```

The following example for IBM Spectrum LSF Application Center (LSF Application Center) considerably reduces the number of events logged, but standard reports will not work after this configuration:

```
ALLOW_EVENT_TYPE=JOB_NEW JOB_STATUS JOB_FINISH2 JOB_START JOB_EXECUTE JOB_EXT_MSG JOB_SIGNAL
JOB_REQUEUE JOB_MODIFY2
JOB_SWITCH METRIC_LOG
```

The following example reduces the number of events logged for LSF Application Center while still allowing standard reports to work:

```
ALLOW_EVENT_TYPE=JOB_NEW JOB_START JOB_START_ACCEPT JOB_STATUS JOB_FORWARD
MIG JOB_ACCEPT JOB_SWITCH JOB_FINISH JOB_MODIFY2 UNFULFILL JOB_SIGACT
JOB_SIGNAL JOB_EXECUTE JOB_REQUEUE JOB_CLEAN JOB_EXCEPTION JOB_EXT_MSG
JOB_ATTN_DATA JOB_CHUNK SBD_UNREPORTED_STATUS PRE_EXEC_START JOB_FORCE
JOB_MOVE JOB_RUN_RUSAGE METRIC_LOG
```

The following example for IBM Spectrum LSF Analytics is the minimum number of events to provide sufficient information for IBM Spectrum LSF Analytics to generate reports:

```
ALLOW_EVENT_TYPE=JOB_NEW JOB_FINISH2 JOB_STARTLIMIT JOB_STATUS2 JOB_PENDING_REASONS
```

AR_AVAILABLE_STATUS

Syntax

```
AR_AVAILABLE_STATUS={ok closed_full closed_Busy closed_Wind ...}
```

Description

This parameter defines which hosts are considered available for the creation of an AR. It is determined by the following default status configurations:

- ok: hosts that have a batch status as 'ok'
- closed_Full: the previous non-AR jobs (if any) will be suspended
- closed_Busy: the previous non-AR jobs (if any) will be suspended
- closed_Wind: the AR jobs will only run when the host's dispatch time aligns with the AR's active window time

The following host-based statuses can also be considered for the parameter provided the lim status of the host is 'ok':

- closed_Adm
- closed_EGO
- closed_LIM
- closed_Lock
- unavailable
- unreach

Default

Set to {ok closed_full closed_Busy closed_Wind} at the time of installation. Otherwise, not defined.

AR_RUNLIMIT_CHECK

Syntax

```
AR_RUNLIMIT_CHECK=y | Y
```

Description

If set, the system will check the expected RUNLIMIT as well as the estimated runtime of the job and compare it with the closed AR's active window time. It will not dispatch the job in an AR if it is expected for the job to run past the end time of the AR. If a job cannot complete before the end of the AR, the job will be put into an ineligible pending state.

If there is a closed AR job pending due to the RUNLIMIT being over the AR's end time, the AR will be given an ineligible pending status. Current explicitly requested AR jobs will also have an ineligible pending status. Once the RUNLIMIT condition is not valid, the job will return to an eligible pending state.

Default

Set to N, to not check the RUNLIMIT while scheduling.

BJOBS_RES_REQ_DISPLAY

Syntax

BJOBS_RES_REQ_DISPLAY=*none* | *brief* | *full*

Description

This parameter lets you control how many levels of resource requirements `bjobs -l` will display. You can set the parameter to one of the following values:

- `none`: `bjobs -l` does not display any level of resource requirement.
- `brief`: `bjobs -l` only displays the combined and effective resource requirements. This would include, for example, the following:

```
RESOURCE REQUIREMENT DETAILS:
```

```
Combined : res_req
```

```
Effective : res_req
```

- `full`: `bjobs -l` displays the job, app, queue, combined and effective resource requirement. This would include, for example, the following:

```
RESOURCE REQUIREMENT DETAILS:
```

```
Job-level : res_req
```

```
App-level : res_req
```

```
Queue-level: res_req
```

```
Combined : res_req
```

```
Effective : res_req
```

Combined resource requirements are the result of `mbatchd` merging job, application, and queue level resource requirements for each job.

Effective resource requirements are resource requirements used by Scheduler to dispatch jobs. Only the rusage can be changed for running jobs (for example, with `bmod -R`).

When the job finishes, the effective `rsrcreq` that the job last used persists in **JOB_FINISH** of `lsb.acct` and **JOB_FINISH2** of `lsb.stream`. If `mbatchd` was restarted, LSF recovers job effective `rsrcreq` with the one persisted in the **JOB_START** event. When replaying the **JOB_EXECUTE** event, job effective `rsrcreq` recovers the effective `rsrcreq` persisted in **JOB_EXECUTE**.

After modifying this parameter, use `badmin reconfig` or `badmin mbdrestart` make the new value take effect.

Default

brief

BSWITCH_MODIFY_RUSAGE

Syntax

BSWITCH_MODIFY_RUSAGE=*Y*|*y*|*N*|*n*

Description

By default, LSF does not modify effective resource requirements and job resource usage when running the `bswitch` command. The effective resource requirement string for scheduled jobs represents the resource requirement used by the scheduler to make a dispatch decision. Enable this parameter to allow `bswitch` to update job resource usage according to the resource requirements in the new queue.

Default

Not defined. **bswitch** does not update effective resource usage of the job.

CHUNK_JOB_DURATION**Syntax**

CHUNK_JOB_DURATION=*minutes*

Description

Specifies a CPU limit, run limit, or estimated run time for jobs submitted to a chunk job queue to be chunked.

When **CHUNK_JOB_DURATION** is set, the CPU limit or run limit set at the queue level (**CPULIMIT** or **RUNLIMIT**), application level (**CPULIMIT** or **RUNLIMIT**), or job level (-c or -W **bsub** options), or the run time estimate set at the application level (**RUNTIME**) must be less than or equal to **CHUNK_JOB_DURATION** for jobs to be chunked.

If **CHUNK_JOB_DURATION** is set, jobs are *not* chunked if:

- No CPU limit, run time limit, or run time estimate is specified at any level, or
- A CPU limit, run time limit, or run time estimate is greater than the value of **CHUNK_JOB_DURATION**.

The value of **CHUNK_JOB_DURATION** is displayed by **bparams -1**.

Examples

- **CHUNK_JOB_DURATION** is not defined:
 - Jobs with no CPU limit, run limit, or run time estimate are chunked
 - Jobs with a CPU limit, run limit, or run time estimate less than or equal to 30 are chunked
 - Jobs with a CPU limit, run limit, or run time estimate greater than 30 are *not* chunked
- **CHUNK_JOB_DURATION=90**:
 - Jobs with no CPU limit, run limit, or run time estimate are *not* chunked
 - Jobs with a CPU limit, run limit, or run time estimate less than or equal to 90 are chunked
 - Jobs with a CPU limit, run limit, or run time estimate greater than 90 are *not* chunked

Default

Not defined.

CLEAN_PERIOD**Syntax**

CLEAN_PERIOD=*seconds*

Description

The amount of time that finished job records are kept in **mbatchd** core memory.

Users can still see all jobs after they have finished using the **bjobs** command.

For jobs that finished more than **CLEAN_PERIOD** seconds ago, use the **bhist** command.

Default

3600 (1 hour)

COMMITTED_RUN_TIME_FACTOR

Syntax

COMMITTED_RUN_TIME_FACTOR=*number*

Description

Used only with fairshare scheduling. Committed run time weighting factor.

In the calculation of a user's dynamic priority, this factor determines the relative importance of the committed run time in the calculation. If the **-W** option of **bsub** is not specified at job submission and a **RUNLIMIT** has not been set for the queue, the committed run time is not considered.

This parameter can also be set for an individual queue in `lsb.queues`. If defined, the queue value takes precedence.

Valid values

Any positive number between 0.0 and 1.0

Default

0.0

CLEAN_PERIOD_DONE

Syntax

CLEAN_PERIOD_DONE=*seconds*

Description

Controls the amount of time during which successfully finished jobs are kept in `mbatchd` core memory. This applies to DONE and PDONE (post job execution processing) jobs.

If **CLEAN_PERIOD_DONE** is not defined, the clean period for DONE jobs is defined by **CLEAN_PERIOD** in `lsb.params`. If **CLEAN_PERIOD_DONE** is defined, its value must be less than **CLEAN_PERIOD**, otherwise it will be ignored and a warning message will appear.

Default

Not defined.

COMPUTE_UNIT_TYPES

Syntax

COMPUTE_UNIT_TYPES=*type1 type2...*

Description

Used to define valid compute unit types for topological resource requirement allocation.

The order in which compute unit types appear specifies the containment relationship between types. Finer grained compute unit types appear first, followed by the coarser grained type that contains them, and so on.

At most one compute unit type in the list can be followed by an exclamation mark designating it as the default compute unit type. If no exclamation mark appears, the first compute unit type in the list is taken as the default type.

Valid values

Any space-separated list of alphanumeric strings.

Default

Not defined

Example

```
COMPUTE_UNIT_TYPES=cell enclosure! rack
```

Specifies three compute unit types, with the default type enclosure. Compute units of type rack contain type enclosure, and of type enclosure contain type cell.

CONDENSE_PENDING_REASONS**Syntax**

```
CONDENSE_PENDING_REASONS=ALL | PARTIAL | N
```

Description

Set to ALL, condenses all host-based pending reasons into one generic pending reason. This is equivalent to setting CONDENSE_PENDING_REASONS=Y.

Set to PARTIAL, condenses all host-based pending reasons except shared resource pending reasons into one generic pending reason.

When the condensed pending reason feature is enabled, only a main reason will be set for a job if it is supposed to have host reasons. In other words, the single key reason feature and the categorized reason feature will show the main reason as "individual host based reasons".

If enabled, you can request a full pending reason list by running the following command:

```
badmin diagnose jobId
```

Tip:

You must be an LSF administrator or a queue administrator to run this command.

When disabled, (CONDENSE_PENDING_REASONS=N), all pending reasons are displayed, including all host-based pending reasons.

Examples

- CONDENSE_PENDING_REASONS=ALL If a job has no other pending reason, `bjobs -p` or `bjobs -l` displays the following:

```
Individual host based reasons
```

- CONDENSE_PENDING_REASONS=N The pending reasons are not condensed or filtered:

Default

Set to N by default.

CPU_TIME_FACTOR**Syntax**

```
CPU_TIME_FACTOR=number
```

Description

Used only with fairshare scheduling. CPU time weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the cumulative CPU time used by a user's jobs.

This parameter can also be set for an individual queue in `lsb.queues`. If defined, the queue value takes precedence.

Default

0.7

CSM_VALID_SMT**Syntax**

`CSM_VALID_SMT=default_smt_value [smt_valid ...]`

Description

Defines a space-delimited list of valid SMT mode values.

This option is for LSF jobs that are submitted with the IBM Cluster Systems Manager (CSM) integration package.

The first value is the default SMT mode for CSM jobs if the SMT mode is not specified at the queue or job level.

Valid values

A space-delimited list of positive integers. Duplicate values are ignored. This value is overwritten by the **smt** value of the **CSM_REQ** parameter in the **lsb.queues** file if it is specified.

Example

```
CSM_VALID_SMT=8 2 4
```

Valid SMT mode values are 2, 4, and 8. The default SMT mode is 8.

Default

0.

DEFAULT_APPLICATION**Syntax**

`DEFAULT_APPLICATION=application_profile_name`

Description

The name of the default application profile. The application profile must already be defined in `lsb.applications`.

When you submit a job to LSF without explicitly specifying an application profile, LSF associates the job with the specified application profile.

Default

Not defined. When a user submits a job without explicitly specifying an application profile, and no default application profile is defined by this parameter, LSF does not associate the job with any application profile.

DEFAULT_BWAIT_TIMEOUT

Specifies the default timeout interval to evaluate the wait conditions in a scheduling session, in minutes.

Syntax

DEFAULT_BWAIT_TIMEOUT=*minutes*

Description

If you do not run the **bwait -t** option, the **bwait** command uses this parameter value as the default.

Valid values

An integer in the range 1 - 525600 (one year).

Default

1440 minutes.

DEFAULT_HOST_SPEC**Syntax**

DEFAULT_HOST_SPEC=*host_name* | *host_model*

Description

The default CPU time normalization host for the cluster.

The CPU factor of the specified host or host model will be used to normalize the CPU time limit of all jobs in the cluster, unless the CPU time normalization host is specified at the queue or job level.

Default

Not defined

DEFAULT_JOB_CWD**Syntax**

DEFAULT_JOB_CWD=*directory*

Description

Cluster-wide current working directory (CWD) for the job. The path can be absolute or relative to the submission directory.

The path can include the following dynamic patterns, which are case sensitive:

- %J - job ID
- %JG - job group (if not specified, it will be ignored)
- %I - job index (default value is 0)
- %EJ - execution job ID
- %EI - execution job index

- %P - project name
- %U - user name
- %G - New user group new for both CWD and job output directory
- %H - first execution host name

Unsupported patterns are treated as text.

LSF only creates the directory if CWD includes dynamic patterns. For example:

```
DEFAULT_JOB_CWD=/scratch/jobcwd/%U/%J_%I
```

The job CWD will be created by LSF before the job starts running based on CWD parameter values. For every job, the CWD is determined by the following sequence:

1. **bsub -cwd**. If not defined, LSF goes to steps 2 and 3.
2. Environment variable **LSB_JOB_CWD**. If not defined, LSF goes to steps 3 and 4.
3. Application profile based **JOB_CWD** parameter. If not defined, LSF goes to step 4.
4. Cluster wide CWD (**DEFAULT_JOB_CWD**). If not defined, it means there is no CWD and the submission directory will be used instead.

DEFAULT_JOB_CWD supports all LSF path conventions such as UNIX, UNC and Windows formats. A mixed UNIX and Windows cluster can specify one path for UNIX and another path for Windows, separated by a pipe character (|).

```
DEFAULT_JOB_CWD=unix_path|windows_path
```

The first part of the path must be for UNIX and the second part must be for Windows. Both paths must be full paths.

Default

Not defined.

DEFAULT_JOB_OUTDIR

Syntax

```
DEFAULT_JOB_OUTDIR=directory
```

Description

Set this parameter for LSF to create a cluster wide output directory for the job. Once set, the system starts using the new directory and always tries to create the directory if it does not exist. The directory path can be absolute or relative to the submission directory with dynamic patterns.

The path can include the following dynamic patterns, which are case sensitive:

- %J - job ID
- %JG - job group (if not specified, it will be ignored)
- %I - job index (default value is 0)
- %EJ - execution job ID
- %EI - execution job index
- %P - project name
- %U - user name
- %G - User group new for the job output directory
- %H - first execution host name

Unsupported patterns are treated as text.

For example:

```
DEFAULT_JOB_OUTDIR=/scratch/%U/%J | \\samba\scratch\%U\%J
```

LSF creates the output directory even if the path does not include dynamic patterns. LSF checks the directories from the beginning of the path. If a directory does not exist, the system tries to create that directory. If it fails to create that directory then the system deletes all created directories and uses the submission directory for output. LSF creates all directories under the 700 permissions with the ownership of a submission user.

DEFAULT_JOB_OUTDIR supports all LSF path conventions such as UNIX, UNC and Windows formats. A mixed UNIX and Windows cluster can specify one path for UNIX and another path for Windows, separated by a pipe character (|).

```
DEFAULT_JOB_OUTDIR=unix_path|windows_path
```

The first part of the path must be for UNIX and the second part must be for Windows. Both paths must be full paths.

An output directory can also be created for a checkpointed job.

Default

Not defined. The system uses the submission directory for job output.

DEFAULT_JOBGROUP

Syntax

```
DEFAULT_JOBGROUP=job_group_name
```

Description

The name of the default job group.

When you submit a job to LSF without explicitly specifying a job group, LSF associates the job with the specified job group. The **LSB_DEFAULT_JOBGROUP** environment variable overrides the setting of **DEFAULT_JOBGROUP**. The **bsub -g *job_group_name*** option overrides both **LSB_DEFAULT_JOBGROUP** and **DEFAULT_JOBGROUP**.

Default job group specification supports macro substitution for project name (%p) and user name (%u). When you specify **bsub -P *project_name***, the value of %p is the specified project name. If you do not specify a project name at job submission, %p is the project name defined by setting the environment variable **LSB_DEFAULTPROJECT**, or the project name specified by **DEFAULT_PROJECT** in `lsb.params`. the default project name is default.

For example, a default job group name specified by **DEFAULT_JOBGROUP=/canada/%p/%u** is expanded to the value for the LSF project name and the user name of the job submission user (for example, /canada/projects/user1).

Job group names must follow this format:

- Job group names must start with a slash character (/). For example, **DEFAULT_JOBGROUP=/A/B/C** is correct, but **DEFAULT_JOBGROUP=A/B/C** is not correct.
- Job group names cannot end with a slash character (/). For example, **DEFAULT_JOBGROUP=/A/** is not correct.
- Job group names cannot contain more than one slash character (/) in a row. For example, job group names like **DEFAULT_JOBGROUP=/A//B** or **DEFAULT_JOBGROUP=A///B** are not correct.
- Job group names cannot contain spaces. For example, **DEFAULT_JOBGROUP=/A/B C/D** is not correct.
- Project names and user names used for macro substitution with %p and %u cannot start or end with slash character (/).

lsb.params

- Project names and user names used for macro substitution with %p and %u cannot contain spaces or more than one slash character (/) in a row.
- Project names or user names containing slash character (/) will create separate job groups. For example, if the project name is canada/projects, DEFAULT_JOBGROUP=/%p results in a job group hierarchy /canada/projects.

Example

```
DEFAULT_JOBGROUP=/canada/projects
```

Default

Not defined. When a user submits a job without explicitly specifying job group name, and the **LSB_DEFAULT_JOBGROUP** environment variable is not defined, LSF does not associate the job with any job group.

DEFAULT_PROJECT

Syntax

```
DEFAULT_PROJECT=project_name
```

Description

The name of the default project. Specify any string.

Project names can be up to 511 characters long.

When you submit a job without specifying any project name, and the environment variable **LSB_DEFAULTPROJECT** is not set, LSF automatically assigns the job to this project.

Default

```
default
```

DEFAULT_QUEUE

Syntax

```
DEFAULT_QUEUE=queue_name ...
```

Description

Space-separated list of candidate default queues (candidates must already be defined in `lsb.queues`).

When you submit a job to LSF without explicitly specifying a queue, and the environment variable **LSB_DEFAULTQUEUE** is not set, LSF puts the job in the first queue in this list that satisfies the job's specifications subject to other restrictions, such as requested hosts, queue status, etc.

Default

This parameter is set at installation to **DEFAULT_QUEUE=normal interactive**.

When a user submits a job to LSF without explicitly specifying a queue, and there are no candidate default queues defined (by this parameter or by the user's environment variable **LSB_DEFAULTQUEUE**), LSF automatically creates a new queue named default, using the default configuration, and submits the job to that queue.

DEFAULT_RESREQ_ORDER

Syntax

DEFAULT_RESREQ_ORDER=*order_string*

Description

The *order_string* is `[!][-]resource_name [:[-]resource_name]...`

Specify the global LSF default sorting order for resource requirements so the scheduler can find the right candidate host. You can specify any built-in or external load index or static resource. When an index name is preceded by a minus sign (-), the sorting order is reversed so that hosts are ordered from worst to best on that index. A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Default

r15s:pg

DEFAULT_SLA_VELOCITY

Syntax

DEFAULT_SLA_VELOCITY=*num_slots*

Description

For EGO-enabled SLA scheduling, the number of slots that the SLA should request for parallel jobs running in the SLA.

By default, an EGO-enabled SLA requests slots from EGO based on the number of jobs the SLA needs to run. If the jobs themselves require more than one slot, they will remain pending. To avoid this for parallel jobs, set **DEFAULT_SLA_VELOCITY** to the total number of slots that are expected to be used by parallel jobs.

Default

1

DEFAULT_USER_GROUP

Syntax

DEFAULT_USER_GROUP=*default_user_group*

Description

When **DEFAULT_USER_GROUP** is defined, all submitted jobs must be associated with a user group. Jobs without a user group specified will be associated with *default_user_group*, where *default_user_group* is a group configured in `lsb.users` and contains all as a direct member. **DEFAULT_USER_GROUP** can only contain one user group.

If the default user group does not have shares assigned in a fairshare queue, jobs can still run from the default user group and are charged to the highest priority account the user can access in the queue. A job submitted to a user group without shares in a specified fairshare queue is transferred to the default user group where the job can run. A job modified or moved using `bmod` or `bswitch` may similarly be transferred to the default user group.

Note:

The default user group should be configured in most queues and have shares in most fairshare queues to ensure jobs run smoothly.

Jobs linked to a user group, either through the *default_user_group* or a user group specified at submission using **bsub -G**, allow the user group administrator to issue job control operations. User group administrator rights are configured in the **UserGroup** section `lsb.users`, under **GROUP_ADMIN**.

When **DEFAULT_USER_GROUP** is not defined, jobs do not require a user group association.

After adding or changing **DEFAULT_USER_GROUP** in `lsb.params`, use `badadmin reconfig` to reconfigure your cluster

Default

Not defined. When a user submits a job without explicitly specifying user group name, LSF does not associate the job with any user group.

See also

STRICT_UG_CONTROL, ENFORCE_ONE_UG_LIMITS

DETECT_IDLE_JOB_AFTER

Syntax

`DETECT_IDLE_JOB_AFTER=time_minutes`

Description

The minimum job run time before `mbatchd` reports that the job is idle.

Default

20 (`mbatchd` checks if the job is idle after 20 minutes of run time)

DIAGNOSE_LOGDIR

Syntax

`DIAGNOSE_LOGDIR=<full directory path>`

Description

Set **DIAGNOSE_LOGDIR** to specify the file location for the collected diagnosis information (query source information in a log file, or scheduling bucket information in a snapshot file).

If you do not modify this parameter, the default location for the log file is `LSF_LOGDIR`.

For query source information, the log file shows who issued these requests, where the requests came from, and the data size of the query. The name of the log file is `query_info.querylog.<host_name>`.

You can dynamically set the path from the command line with **badadmin diagnose -c query -f log_name**, where `log_name` can be a full path. This overrides any other setting for the path. However, if you restart/reconfigure `mbatchd`, this path setting is lost and it defaults back to the setting you specified in this parameter.

For example, the output of the log file is:

```
Jan 23 15:58:12 2020 bqueues,root,sjc-host10.rtp.stglabs.ibm.com,25212,0x1
```

Where:

- The 1st field is the timestamp. The delimiter between the 1st and 2nd field is a space.

- The 2nd field is the trigger query command. Only mbatchd query commands are supported. Some commands may trigger multiple queries/entries.
- The 3rd field is the user. If the user name is unknown, LSF displays -.
- The 4th field is the sending host. If the host name cannot be resolved, LSF uses the IP if available, or -.
- The 5th field is the maximum data size in bytes. The real data size may be significantly larger or smaller.
- The 6th field is for query options. Check lsbatch.h for details. If not applied, LSF displays -.

The values are separated by commas to make it easier to write a script for analyzing the data.

For scheduler job bucket information, the snapshot file contains an active image of scheduler job buckets as raw data in XML or JSON format. By default, the name of the snapshot file is `jobreq_<hostname>_<dateandtime>.<format>`, and is in XML format.

You can dynamically set the path from the command line with **badmin diagnose -c jobreq -f snapshot_name**, where `snapshot_name` can be a full path, and use `-t json` to specify that the snapshot file is in JSON format. This overrides any other setting for the path.

Default

LSF_LOGDIR/

DISABLE_UACCT_MAP

Syntax

DISABLE_UACCT_MAP=y | Y

Description

Specify y or Y to disable user-level account mapping.

Default

N

DOCKER_IMAGE_AFFINITY

Syntax

DOCKER_IMAGE_AFFINITY=Y | y | N | n

Description

When scheduling Docker-based containerized jobs, setting this parameter to y or Y enables LSF to give preference for execution hosts that already have the requested Docker image. This reduces network bandwidth and the job start time because the execution host does not have to pull the Docker image from the repository and the job can immediately start on the execution host.

When this feature is enabled, LSF considers Docker image location information when scheduling Docker jobs. Docker image affinity interacts with host preference and **order[]** string requests in the following manner:

- If host preference is specified, the host preference is honored first. Among hosts with the same preference level, hosts with the requested Docker image are given higher priority.
- If the **order[]** string is specified, the hosts with the requested Docker image have a higher priority first. Among hosts that all have the requested Docker image, the **order[]** string is then honored.

Default

N.

EADMIN_TRIGGER_DURATION

Syntax

EADMIN_TRIGGER_DURATION=*minutes*

Description

Defines how often LSF_SERVERDIR/eadmin is invoked once a job exception is detected. Used in conjunction with job exception handling parameters **JOB_IDLE**, **JOB_OVERRUN**, and **JOB_UNDERRUN** in lsb.queues.

When there are exceptions in updating the stream file, LSF_SERVERDIR/eadmin also runs to report the stream file exceptions.

Tip:

Tune EADMIN_TRIGGER_DURATION carefully. Shorter values may raise false alarms, longer values may not trigger exceptions frequently enough.

Example

```
EADMIN_TRIGGER_DURATION=5
```

Default

1 minute

EADMIN_TRIGGER_INTERVAL

Syntax

EADMIN_TRIGGER_INTERVAL=*seconds*

Description

When set to an integer greater than 0, this parameter defines how often the eadmin script found in LSF_SERVERDIR is invoked by mbatchd, even without a job exception.

Tip:

Tune EADMIN_TRIGGER_INTERVAL carefully. Shorter values may raise false alarms, longer values may not trigger exceptions frequently enough.

Example

```
EADMIN_TRIGGER_INTERVAL=3600
```

Default

0 : The eadmin script is triggered according to the **EADMIN_TRIGGER_DURATION** parameter value.

EGO_SLOTBASED_VELOCITY_SLA

Syntax

EGO_SLOTBASED_VELOCITY_SLA=Y|N

Description

Enables slot based requirements for EGO-enabled SLA. If the value is N, LSF calculates how many slots you need by the number of jobs. If the value is Y, LSF calculates how many slots you need by the number of job slots instead of the number of jobs.

Default

Y

EGROUP_UPDATE_INTERVAL

Specifies the time interval for which dynamic user and host group information is updated automatically.

Syntax

EGROUP_UPDATE_INTERVAL=*hours* | *minutes* *m*

Description

Specify a time interval for which dynamic user group information in the `lsb.users` file and dynamic host group information in the `lsb.hosts` file is updated automatically. Specify the time interval in hours or use the *m* suffix to specify the time interval in minutes. There is no need to run **badmin reconfig**.

In the LSF_SERVERDIR, there must be an executable file named **egroup** to manage the host and user group members. When **EGROUP_UPDATE_INTERVAL** is set, and the time interval is matched, LSF gets the updated members from **egroup**.

If this parameter is not set, you must update the host and user groups manually by running the **badmin reconfig** command.

Default

Not defined.

ELIGIBLE_PENDINFO_SNAPSHOT_INTERVAL

Specifies the time interval for **mbschd** to dump eligible and ineligible pending information to disk.

Syntax

ELIGIBLE_PENDINFO_SNAPSHOT_INTERVAL=*minutes*

Description

If **TRACK_ELIGIBLE_PENDINFO** is enabled, this parameter allows the eligible and ineligible pending job data to be saved to disk so that it is not lost when **mbatchd** restarts. Specify the time interval, in minutes, for **mbschd** to save eligible and ineligible pending data to disk.

Default

If **TRACK_ELIGIBLE_PENDINFO** is enabled, the default is 5 minutes. Otherwise, this parameter is ignored.

See also

TRACK_ELIGIBLE_PENDINFO

ENABLE_DEFAULT_EGO_SLA

Syntax

ENABLE_DEFAULT_EGO_SLA=*service_class_name* | *consumer_name*

Description

The name of the default service class or EGO consumer name for EGO-enabled SLA scheduling. If the specified SLA does not exist in `lsb.serviceclasses`, LSF creates one with the specified consumer name, velocity of 1, priority of 1, and a time window that is always open.

If the name of the default SLA is not configured in `lsb.serviceclasses`, it must be the name of a valid EGO consumer.

ENABLE_DEFAULT_EGO_SLA is required to turn on EGO-enabled SLA scheduling. All LSF resource management is delegated to EGO, and all LSF hosts are under EGO control. When all jobs running in the default SLA finish, all allocated hosts are released to EGO after the default idle timeout of 120 seconds (configurable by **MAX_HOST_IDLE_TIME** in `lsb.serviceclasses`).

When you submit a job to LSF without explicitly using the `-sla` option to specify a service class name, LSF puts the job in the default service class specified by *service_class_name*.

Default

Not defined. When a user submits a job to LSF without explicitly specifying a service class, and there is no default service class defined by this parameter, LSF does not attach the job to any service class.

ENABLE_DIAGNOSE

Syntax

ENABLE_DIAGNOSE=query

Description

Enable this parameter for `mbatchd` to write query source information to a log file (see `DIAGNOSE_LOGDIR` in `lsb.params`). The log file shows information about the source of `mbatchd` queries, allowing you to troubleshoot problems. The log file shows who issued these requests, where the requests came from, and the data size of the query.

The log file collects key information like query name, user name, host name and the data size of the query. You can write a script to format the output.

Default

Disabled

ENABLE_EVENT_STREAM

Syntax

ENABLE_EVENT_STREAM=Y | N

Description

Used only with event streaming for system performance analysis tools.

Default

N (event streaming is not enabled)

ENABLE_EXIT_RATE_PER_SLOT

Syntax

ENABLE_EXIT_RATE_PER_SLOT=Y | N

Description

Scales the actual exit rate thresholds on a host according to the number of slots on the host. For example, if EXIT_RATE=2 in `lsb.hosts` or GLOBAL_EXIT_RATE=2 in `lsb.params`, and the host has 2 job slots, the job exit rate threshold will be 4.

Default

N

ENABLE_GPU_HIST_RUN_TIME

Enables the use of historical GPU run time in the calculation of fairshare scheduling priority. Used with fairshare scheduling.

Syntax

ENABLE_GPU_HIST_RUN_TIME=y | Y | n | N

Description

If set to Y, enables the use of historical GPU run time in the calculation of fairshare scheduling priority.

This parameter can also be set for an individual queue in `lsb.queues`. If defined, the queue value takes precedence.

Default

N

ENABLE_HIST_RUN_TIME

Syntax

ENABLE_HIST_RUN_TIME=y | Y | n | N

Description

Used with fairshare scheduling and global fairshare scheduling. If set, enables the use of historical run time in the calculation of fairshare scheduling priority.

Whether or not **ENABLE_HIST_RUN_TIME** is set for a global fairshare queue, the historical run time for share accounts in the global fairshare queue is reported to GPD. When GPD receives historical run time from one cluster, it broadcasts the historical run time to other clusters. The remote historical run time received from GPD is not used in the calculation for fairshare scheduling priority for the queue.

This parameter can also be set for an individual queue in `lsb.queues`. If defined, the queue value takes precedence.

Default

N

ENABLE_HOST_INTERSECTION

Syntax

ENABLE_HOST_INTERSECTION=Y | N

Description

When enabled, allows job submission to any host that belongs to the intersection created when considering the queue the job was submitted to, any advance reservation hosts, or any hosts specified by **bsub -m** at the time of submission.

When disabled job submission with hosts specified can be accepted only if specified hosts are a subset of hosts defined in the queue.

The following commands are affected by **ENABLE_HOST_INTERSECTION**:

- **bsub**
- **bmod**
- **bmig**
- **brestart**
- **bswitch**

If no hosts exist in the intersection, the job is rejected.

Default

N

ENABLE_JOB_INFO_BY_ADMIN_ROLE

Syntax

ENABLE_JOB_INFO_BY_ADMIN_ROLE = [usergroup] [queue] [cluster]

Description

By default, an administrator's access to job details is determined by the setting of **SECURE_JOB_INFO_LEVEL**, the same as a regular user. The parameter **ENABLE_JOB_INFO_BY_ADMIN_ROLE** in lsb.params allows you to enable the user group, queue, and cluster administrators the right to access job detail information for jobs in the user group, queue, and clusters they manage, even when the administrator has no right based on the configuration of **SECURE_JOB_INFO_LEVEL**.

You may define one or more of the values, usergroup, queue, or cluster.

Default

NULL (Not defined)

ENABLE_USER_RESUME

Syntax

ENABLE_USER_RESUME=Y | N

Description

Defines job resume permissions.

When this parameter is defined:

- If the value is Y, users can resume their own jobs that have been suspended by the administrator.
- If the value is N, jobs that are suspended by the administrator can only be resumed by the administrator or root; users do not have permission to resume a job suspended by another user or the administrator. Administrators can resume jobs suspended by users or administrators.

Default

N (users cannot resume jobs suspended by administrator)

ENFORCE_ONE_UG_LIMITS

Syntax

ENFORCE_ONE_UG_LIMITS=Y | N

Upon job submission with the -G option and when user groups have overlapping members, defines whether only the specified user group's limits (or those of any parent group) are enforced or whether the most restrictive user group limits of any overlapping user/user group are enforced.

- If the value is Y, only the limits defined for the user group that you specify with the -G option during job submission apply to the job, even if there are overlapping members of groups. Resource limits apply to all user groups to which you belong except for any user groups that you exclude with the -G option (-G "~group_name").

If you have nested user groups, the limits of a user's group parent also apply.

View the jobs under the user groups that you specified with -G by running **busers**. **busers** shows the jobs under the specified user group. If you have nested user groups, the jobs also appear under the user group's parent groups. If you did not specify a user group with -G, the jobs appear under all user groups to which you belong. If you exclude a user group in the job submission (-G "~group_name"), those jobs do not appear under that user group.

View existing limits by running **blimits**.

- If the value is N and the user group has members that overlap with other user groups, the strictest possible limits (that you can view by running **blimits**) defined for any of the member user groups are enforced for the job. Resource limits apply to all user groups to which you belong, even to user groups that you exclude with the -G option (-G "~group_name").

If the user group specified at submission is no longer valid when the job runs and ENFORCE_ONE_UG_LIMIT=Y, only the user limit is applied to the job. This can occur if the user group is deleted or the user is removed from the user group.

Examples

For a cluster with the following resource limits defined in the `lsb.resources` file:

```
Begin Limit
NAME = limit1
USERS = group1
PER_HOST = all
SLOTS = 2
End Limit

Begin Limit
NAME = limit2
USERS = group2
ELIGIBLE_PEND_JOBS = 2
JOBS = 5
End Limit

Begin Limit
NAME = limit3
USERS = group3
ELIGIBLE_PEND_JOBS = 2
JOBS = 5
End Limit
```

lsb.params

If ENFORCE_ONE_UG_LIMIT=Y is enabled and user1 belongs to the group1, group2, and group3 user groups, the following commands have the following effects:

- **bsub -G "group1"**

Limits apply only to group1 and its parents. The limit1 limit applies.

- **bsub -G "~group1 ~group2"**

Limits do not apply to group1 and group2. The limit3 limit applies.

Default

N

ENFORCE_UG_TREE

Syntax

ENFORCE_UG_TREE=Y | N

Description

When ENFORCE_UG_TREE=Y is defined, user groups must form a tree-like structure, with each user group having at most one parent. User group definitions in the **UserGroup** section of lsb.users will be checked in configuration order, and any user group appearing in **GROUP_MEMBER** more than once will be ignored after the first occurrence.

After adding or changing **ENFORCE_UG_TREE** in lsb.params, use badadmin reconfig to reconfigure your cluster

Default

N (Not defined.)

See also

DEFAULT_USER_GROUP, ENFORCE_ONE_UG_LIMIT, STRICT_UG_CONTROL

ESTIMATED_RUNTIME

Syntax

ESTIMATED_RUNTIME=[hour:]minute[/host_name | /host_model]

Description

This parameter specifies an estimated run time for jobs. LSF uses the ESTIMATED_RUNTIME value for scheduling purposes only, and does not kill jobs that exceed this value unless the jobs also exceed a defined RUNLIMIT. The format of runtime estimate is same as the RUNLIMIT parameter.

The job-level runtime estimate specified by **bsub -We**, the ESTIMATED_RUNTIME setting in an application profile, or the ESTIMATED_RUNTIME setting in a queue all override the cluster-wide ESTIMATED_RUNTIME setting.

The following LSF features use the ESTIMATED_RUNTIME value to schedule jobs:

- Job chunking
- Advance reservation
- SLA
- Slot reservation
- Backfill

- Allocation planner

Default

Not defined

ESTIMATOR_CPUS

Syntax

```
ESTIMATOR_CPUS=cpu_num [ cpu_num ...]
```

Description

Defines a space-delimited list of master host CPU numbers where the job start time estimator processes can run.

LSF uses separate processes for the simulation-based job start time estimator. Since these processes are running compute-intensive tasks on the master host, job start time estimation can impact the scheduling performance of **mbatchd** and **mbschd**. Use this parameter to bind the estimator processes to the specified CPUs, which reduces the impact of the estimator on the performance of **mbatchd** and **mbschd**.

You must also define `LSB_ENABLE_ESTIMATION=Y` in `lsf.conf` to enable job start time estimation and to use this parameter.

Valid values

A space-delimited list of CPU numbers.

Example

```
ESTIMATOR_CPUS=1 2
```

The estimator processes can only run on CPUs number 1 and 2.

Default

Not defined. The estimator processes can run on any CPUs on the master host.

See also

ESTIMATOR_SIM_START_INTERVAL, **ESTIMATOR_MAX_TIME_PREDICTION**, **ESTIMATOR_MAX_JOBS_PREDICTION**, and **ESTIMATOR_MAX_RUNTIME_PREDICTION** parameters.

LSB_ENABLE_ESTIMATION parameter in `lsf.conf`.

ESTIMATOR_MAX_JOBS_PREDICTION

Syntax

```
ESTIMATOR_MAX_JOBS_PREDICTION=max_num_jobs
```

Description

Specifies the maximum number of jobs that the job start time estimator can predict. **mbatchd** stops the current round of estimation when the number of pending jobs dispatched to the estimator reaches this number, or if there are no more pending jobs to estimate. Use this parameter to limit the number of pending jobs for which LSF will provide start time estimates using simulation-based estimation.

LSF stops the current round of estimation when the estimator reaches any one of the estimation thresholds (**ESTIMATOR_MAX_JOBS_PREDICTION**, **ESTIMATOR_MAX_RUNTIME_PREDICTION**, and **ESTIMATOR_MAX_TIME_PREDICTION**).

You must also define `LSB_ENABLE_ESTIMATION=Y` in `lsf.conf` to enable job start time estimation and to use this parameter.

Valid values

Any integer greater than zero.

Default

1000 jobs.

See also

ESTIMATOR_SIM_START_INTERVAL, **ESTIMATOR_MAX_TIME_PREDICTION**, **ESTIMATOR_MAX_RUNTIME_PREDICTION**, and **ESTIMATOR_CPUS** parameters.

LSB_ENABLE_ESTIMATION parameter in `lsf.conf`.

ESTIMATOR_MAX_RUNTIME_PREDICTION

Syntax

`ESTIMATOR_MAX_RUNTIME_PREDICTION=minutes`

Description

Specifies the maximum time, in minutes, that the job start time estimator can run for each round. Once the run time of the estimator reaches this threshold, or if there are no more pending jobs to estimate, **mbatchd** stops the current round of estimation after the estimator reports all the estimated job start times back to **mbatchd**. Use this parameter to limit the amount of time that LSF uses to run the estimator.

LSF stops the current round of estimation when the estimator reaches any one of the estimation thresholds (**ESTIMATOR_MAX_JOBS_PREDICTION**, **ESTIMATOR_MAX_RUNTIME_PREDICTION**, and **ESTIMATOR_MAX_TIME_PREDICTION**).

You must also define `LSB_ENABLE_ESTIMATION=Y` in `lsf.conf` to enable job start time estimation and to use this parameter.

Valid values

Any integer from 1 to **ESTIMATOR_SIM_START_INTERVAL**. If set to a larger value, LSF defaults to the value of **ESTIMATOR_SIM_START_INTERVAL**.

Default

30. The estimator can run up to 30 minutes, or for the amount of time as specified by the **ESTIMATOR_SIM_START_INTERVAL** parameter, whichever is smaller.

See also

ESTIMATOR_SIM_START_INTERVAL, **ESTIMATOR_MAX_TIME_PREDICTION**, **ESTIMATOR_MAX_JOBS_PREDICTION**, and **ESTIMATOR_CPUS** parameters.

LSB_ENABLE_ESTIMATION parameter in `lsf.conf`.

ESTIMATOR_MAX_TIME_PREDICTION

Syntax

`ESTIMATOR_MAX_TIME_PREDICTION=minutes`

Description

Specifies the maximum amount of time into the future, in minutes, for which the job start time estimator predicts the start time of jobs. **mbatchd** stops the current round of estimation when the start time prediction of a job reaches this value. Use this parameter to control how far into the future that LSF can predict as the start time of jobs using simulation-based estimation.

Each round of job start time estimation uses a snapshot of the LSF environment at that time to predict the start time of pending jobs in the cluster. The LSF environment (including the running jobs and available resources in the cluster) may change between each round of estimation. This means that the start time prediction for jobs with lower priorities, which usually start further into the future, might change as higher-priority jobs are submitted to LSF.

Decreasing the value of this parameter provides estimates for a fewer number of jobs, but the estimates are more meaningful because the provided estimates are less likely to change over time. Decreasing the value of this parameter also decreases the amount of time required to finish one round of estimation.

Conversely, increasing the value of this parameter provides start time estimates for more jobs, but the accuracy diminishes as the estimates are provided further into the future. Increasing the value of this parameter also increases the amount of time it takes to finish one round of estimation.

LSF stops the current round of estimation when the estimator reaches any one of the estimation thresholds (**ESTIMATOR_MAX_JOBS_PREDICTION**, **ESTIMATOR_MAX_RUNTIME_PREDICTION**, and **ESTIMATOR_MAX_TIME_PREDICTION**).

You must also define `LSB_ENABLE_ESTIMATION=Y` in `lsf.conf` to enable job start time estimation and to use this parameter.

Valid values

Any integer from 1 to **ESTIMATOR_SIM_START_INTERVAL**.

Default

10080 minutes. LSF stops the estimation round once a job start time is predicted to be one week into the future.

See also

ESTIMATOR_SIM_START_INTERVAL, **ESTIMATOR_MAX_JOBS_PREDICTION**, **ESTIMATOR_MAX_RUNTIME_PREDICTION**, and **ESTIMATOR_CPUS** parameters.

LSB_ENABLE_ESTIMATION parameter in `lsf.conf`.

ESTIMATOR_SIM_START_INTERVAL

Syntax

`ESTIMATOR_SIM_START_INTERVAL=minutes`

Description

Specifies the amount of time, in minutes, since the start of the last round of job start time estimation that **mbatchd** waits before triggering another round of estimation. Use this parameter to control how often LSF uses simulation-based estimation to predict the start time of jobs.

Each round of job start time estimation uses a snapshot of the LSF environment at that time to predict the start time of pending jobs in the cluster. The LSF environment (including the running jobs and available resources in the cluster) may change between each round of estimation, so decreasing the value of this parameter increases the prediction accuracy, but at the expense of the computation effort.

Conversely, increasing the value of this parameter decreases the prediction accuracy while decreasing the computation effort. If your LSF environment does not change frequently and there are the same jobs

pending in each subsequent round of estimation, the predicted job start times for the same pending jobs should also not change between estimation rounds. Therefore, increasing the value of this parameter can avoid redundant estimation rounds for long-pending jobs.

You must also define `LSB_ENABLE_ESTIMATION=Y` in `lsf.conf` to enable job start time estimation and to use this parameter.

Valid values

Any integer greater than zero.

Default

30 minutes.

See also

ESTIMATOR_MAX_TIME_PREDICTION, **ESTIMATOR_MAX_JOBS_PREDICTION**, **ESTIMATOR_MAX_RUNTIME_PREDICTION**, and **ESTIMATOR_CPUS** parameters.

LSB_ENABLE_ESTIMATION parameter in `lsf.conf`.

EVALUATE_JOB_DEPENDENCY

Set the maximum number of job dependencies **mbatchd** evaluates in one scheduling cycle.

Syntax

`EVALUATE_JOB_DEPENDENCY=integer`

Description

This parameter limits the amount of time **mbatchd** spends on evaluating job dependencies in a scheduling cycle, which limits the amount of time the job dependency evaluation blocks services. Job dependency evaluation is a process that is used to check whether each job's dependency condition is satisfied. When a job's dependency condition is satisfied, it sets a ready flag and allows itself to be scheduled by **mbschd**.

When **EVALUATE_JOB_DEPENDENCY** is set, a configured number of jobs are evaluated. Not all the dependency satisfied jobs can be set to READY status in the same session. Therefore, jobs intended to be dispatched in one scheduling session might be dispatched in different scheduling sessions.

Also, the job dependency evaluation process starts from the last evaluation end location. Therefore, LSF might prevent some dependency satisfied jobs that occur before the end location from being set to READY status in that particular session. As a result, one job might be dispatched before another when the other was ready first. LSF starts the job dependency evaluation from the endpoint in the next session. LSF evaluates all dependent jobs every 10 minutes regardless of the configuration for **EVALUATE_JOB_DEPENDENCY**.

The **EVALUATE_JOB_DEPENDENCY_TIMEOUT** parameter provides an explicit time limit on how long **mbatchd** spends on evaluating job dependencies in one scheduling cycle, in seconds or milliseconds. If you specify values for both the **EVALUATE_JOB_DEPENDENCY** and **EVALUATE_JOB_DEPENDENCY_TIMEOUT** parameters, LSF ignores the **EVALUATE_JOB_DEPENDENCY** parameter value and instead uses the value of the **EVALUATE_JOB_DEPENDENCY_TIMEOUT** parameter. If you specify a value for the **EVALUATE_JOB_DEPENDENCY** parameter but not the **EVALUATE_JOB_DEPENDENCY_TIMEOUT** parameter, the **EVALUATE_JOB_DEPENDENCY** parameter takes effect.

Default

Unlimited.

Related referenceEVALUATE_JOB_DEPENDENCY_TIMEOUT

Set the maximum amount of time, in seconds or milliseconds, that the **mbatchd** daemon takes to evaluate job dependencies in one scheduling cycle.

EVALUATE_JOB_DEPENDENCY_TIMEOUT

Set the maximum amount of time, in seconds or milliseconds, that the **mbatchd** daemon takes to evaluate job dependencies in one scheduling cycle.

Syntax

`EVALUATE_JOB_DEPENDENCY_TIMEOUT=time_in_seconds`

`EVALUATE_JOB_DEPENDENCY_TIMEOUT=time_in_milliseconds ms`

Description

This parameter limits the amount of time that **mbatchd** spends on evaluating job dependencies in a scheduling cycle, which limits the amount of time the job dependency evaluation blocks services. Job dependency evaluation is a process that is used to check whether each job's dependency condition is satisfied. When a job's dependency condition is satisfied, it sets a ready flag and allows itself to be scheduled by **mbschd**.

When **EVALUATE_JOB_DEPENDENCY_TIMEOUT** is set, **mbatchd** spends up to this much time, in seconds or milliseconds, evaluating job dependency conditions in one scheduling session. Use the `ms` keyword if you are specifying the time limit in milliseconds.

Not all the dependency satisfied jobs can be set to READY status in the same session. Therefore, jobs intended to be dispatched in one scheduling session might be dispatched in different scheduling sessions. Also, the job dependency evaluation process starts from the last evaluation end location. Therefore, LSF might prevent some dependency satisfied jobs that occur before the end location from being set to READY status in that particular session. As a result, one job might be dispatched before another when the other was ready first. LSF starts the job dependency evaluation from the endpoint in the next session. LSF evaluates all dependent jobs every 10 minutes regardless of the configuration for **EVALUATE_JOB_DEPENDENCY_TIMEOUT**.

While the **EVALUATE_JOB_DEPENDENCY** parameter also limits the amount of time that **mbatchd** spends on evaluating job dependency conditions by specifying the maximum number of jobs that are evaluated in one scheduling cycle, the **EVALUATE_JOB_DEPENDENCY_TIMEOUT** parameter provides an explicit time limit. If you specify values for both the **EVALUATE_JOB_DEPENDENCY** and **EVALUATE_JOB_DEPENDENCY_TIMEOUT** parameters, LSF ignores the **EVALUATE_JOB_DEPENDENCY** parameter value and instead uses the value of the **EVALUATE_JOB_DEPENDENCY_TIMEOUT** parameter. If you specify a value for the **EVALUATE_JOB_DEPENDENCY** parameter but not the **EVALUATE_JOB_DEPENDENCY_TIMEOUT** parameter, the **EVALUATE_JOB_DEPENDENCY** parameter takes effect.

Default

1 second.

Related referenceEVALUATE_JOB_DEPENDENCY

Set the maximum number of job dependencies **mbatchd** evaluates in one scheduling cycle.

EVALUATE_WAIT_CONDITION_TIMEOUT

Specifies a limit to the amount of time that the **mbatchd** daemon spends on evaluating the **bwait** wait conditions in a scheduling session.

Syntax

EVALUATE_WAIT_CONDITION_TIMEOUT=*seconds*

EVALUATE_WAIT_CONDITION_TIMEOUT=*milliseconds* ms

Description

If the **mbatchd** daemon cannot evaluate all the wait conditions within this time, the wait conditions might be delayed to the next scheduling session.

To ensure that the wait conditions evaluations are not delayed for too long, the **mbatchd** daemon evaluates all wait conditions every ten minutes regardless of this parameter value.

Valid values

An integer in the range 1 - 1000000.

Default

200 ms.

EVENT_STREAM_FILE

Syntax

EVENT_STREAM_FILE=*file_path*

Description

Determines the path to the event data stream file used by system performance analysis tools.

Default

LSF_TOP/work/*cluster_name*/logdir/stream/lsb.stream

EVENT_UPDATE_INTERVAL

Syntax

EVENT_UPDATE_INTERVAL=*seconds*

Description

Used with duplicate logging of event and accounting log files. **LSB_LOCALDIR** in `lsf.conf` must also be specified. Specifies how often to back up the data and synchronize the directories (**LSB_SHAREDIR** and **LSB_LOCALDIR**).

If you do not define this parameter, the directories are synchronized when data is logged to the files, or when **mbatchd** is started on the first LSF master host. If you define this parameter, **mbatchd** synchronizes the directories only at the specified time intervals.

Use this parameter if NFS traffic is too high and you want to reduce network traffic.

Valid values

1 to 2147483647

Recommended values

Between 10 and 30 seconds, or longer depending on the amount of network traffic.

Note:

Avoid setting the value to exactly 30 seconds, because this will trigger the default behavior and cause mbatchd to synchronize the data every time an event is logged.

Default

Not defined.

See also**LSB_LOCALDIR** in `lsf.conf`**EXIT_RATE_TYPE****Syntax**

EXIT_RATE_TYPE=[JOBEXIT | JOBEXIT_NONLSF] [JOBINIT] [HPCINIT]

Description

When host exception handling is configured (**EXIT_RATE** in `lsb.hosts` or **GLOBAL_EXIT_RATE** in `lsb.params`), specifies the type of job exit to be handled.

JOBEXIT

Job exited after it was dispatched and started running.

JOBEXIT_NONLSF

Job exited with exit reasons related to LSF and not related to a host problem (for example, user action or LSF policy). These jobs are not counted in the exit rate calculation for the host.

JOBINIT

Job exited during initialization because of an execution environment problem. The job did not actually start running.

HPCINIT

HPC job exited during initialization because of an execution environment problem. The job did not actually start running.

Default**JOBEXIT_NONLSF****EXTEND_JOB_EXCEPTION_NOTIFY****Syntax**

EXTEND_JOB_EXCEPTION_NOTIFY=Y | y | N | n

Description

Sends extended information about a job exception in a notification email sent when a job exception occurs. Extended information includes:

lsb.params

- JOB_ID
- RUN_TIME
- IDLE_FACTOR (Only applicable if the job has been idle.)
- USER
- QUEUE
- EXEC_HOST
- JOB_NAME

You can also set format options of the email in the `eadmin` script, located in the `LSF_SERVERDIR` directory. Valid values are **fixed** or **full**.

Default

N (Notification for job exception is standard and includes only job ID and either run time or idle factor.)

FAIRSHARE_ADJUSTMENT_FACTOR

Syntax

FAIRSHARE_ADJUSTMENT_FACTOR=*number*

Description

Used only with fairshare scheduling. Fairshare adjustment plugin weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the user-defined adjustment made in the fairshare plugin (`libfairshareadjust.*`).

A positive float number both enables the fairshare plugin and acts as a weighting factor.

This parameter can also be set for an individual queue in `lsb . queues`. If defined, the queue value takes precedence.

Default

0 (user-defined adjustment made in the fairshare plugin not used)

FWD_JOB_FACTOR

Syntax

FWD_JOB_FACTOR=*number*

Description

Used only with fairshare scheduling. Forwarded job slots weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the number of forwarded job slots reserved and in use by a user when using the LSF multicluster capability.

If enabling fairshare scheduling for the LSF multicluster capability, set this parameter to the same value as **RUN_JOB_FACTOR** unless you want a different weighting factor for forwarded jobs.

This parameter can also be set for an individual queue in `lsb . queues`. If defined, the queue value takes precedence.

Default

Not defined.

See also**RUN_JOB_FACTOR****GLOBAL_EXIT_RATE****Syntax**GLOBAL_EXIT_RATE=*number***Description**

Specifies a cluster-wide threshold for exited jobs. Specify a number of jobs. If **EXIT_RATE** is not specified for the host in `lsb.hosts`, **GLOBAL_EXIT_RATE** defines a default exit rate for all hosts in the cluster. Host-level **EXIT_RATE** overrides the **GLOBAL_EXIT_RATE** value.

If the number of jobs that exit over the period of time specified by **JOB_EXIT_RATE_DURATION** (5 minutes by default) exceeds the number of jobs that you specify as the threshold in this parameter, LSF invokes `LSF_SERVERDIR/eadmin` to trigger a host exception.

Example

GLOBAL_EXIT_RATE=10 defines a job exit rate of 10 jobs for all hosts.

Default

2147483647 (Unlimited threshold.)

GPU_REQ_MERGE**Syntax**

GPU_REQ_MERGE=y | Y | n | N

Description

If set to Y, this parameter enables all individual options in the GPU requirement string to be merged separately. Any specified options override the any options that are specified at the lower levels of precedence. If an individual option is not specified, but is explicitly specified at a lower level, then the highest level for which the option is specified takes precedence. `LSB_GPU_NEW_SYNTAX=extend` must be configured in the `lsb.conf` file for this parameter to take effect.

If this parameter is set to N, the entire GPU requirement string overrides the GPU requirement strings at the lower levels of precedence. Even if an individual option is not specified in the higher level GPU requirement string, the default value of the higher level GPU requirement string takes precedence.

In order of precedence, the job level (**bsub -gpu** command option) takes precedence over the application level (**GPU_REQ** parameter in the `lsb.applications` file), which takes precedence over the queue level (**GPU_REQ** parameter in the `lsb.queues` file).

Example

If the GPU requirements are specified as follows:

- In the `lsb.queues` file for the queueA queue:

```
GPU_REQ="num=2:gmodel=K80"
```

- In the `lsb.applications` file for the appA application profile:

```
GPU_REQ="num=3:nvlink=yes:mode=exclusive_process"
```

- At the job submission level:

```
bsub -gpu "num=1:gmem=10:mode=shared" -q queueA -app appA my_gpu_job
```

If **GPU_REQ_MERGE** is set to Y, LSF merges the individual options separately into the following GPU requirements: "num=1:gmem=10:mode=shared:nvlink=yes:gmodel=K80"

If **GPU_REQ_MERGE** is set to N, the job level GPU requirements take precedence, even for options that are not specified, and LSF uses the following job level GPU requirements:
"num=1:gmem=10:mode=shared"

Note: When there are compound resource requirements that are defined at the job, application, and queue level, the sub-options for the `ngpus_physical` are merged separately if possible, otherwise, the entire **ngpus_physical** specification overrides those of the lower levels of precedence.

Default

N. The entire GPU requirement string overrides the GPU requirement strings at lower levels of precedence.

GPU_RUN_TIME_FACTOR

GPU run time weighting factor. Used only with fairshare scheduling.

Syntax

GPU_RUN_TIME_FACTOR=*number*

Description

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the total GPU run time of a user's running GPU jobs.

This parameter can also be set for an individual queue in `lsb.queues`. If defined, the queue value takes precedence.

Default

0

HIST_HOURS

Syntax

HIST_HOURS=*hours*

Description

Used only with fairshare scheduling. Determines a rate of decay for cumulative CPU time, run time, and historical run time.

To calculate dynamic user priority, LSF scales the actual CPU time and the run time using a decay factor, so that 1 hour of recently-used time is equivalent to 0.1 hours after the specified number of hours has elapsed.

To calculate dynamic user priority with decayed run time and historical run time, LSF scales the accumulated run time of finished jobs and run time of running jobs using the same decay factor, so that 1 hour of recently-used time is equivalent to 0.1 hours after the specified number of hours has elapsed.

When HIST_HOURS=0, CPU time and run time accumulated by running jobs is not decayed.

This parameter can also be set for an individual queue in `lsb.queues`. If defined, the queue value takes precedence.

Default

5

JOB_ACCEPT_INTERVAL**Syntax**JOB_ACCEPT_INTERVAL=*integer*[S | s | M | m]

where S | s indicates that the value is in seconds and M | m indicates that the value is in minutes.

Description

If you set a unit (seconds or minutes), the value that you specify determines how long to wait after dispatching a job to a host before dispatching a second job to the same host.

If you do not set a unit, the value you specify is multiplied by the value of `lsb.params` **MBD_SLEEP_TIME** (60 seconds by default). The result of the calculation is the number of seconds to wait after dispatching a job to a host, before dispatching a second job to the same host.

If 0 (zero), a host may accept more than one job. By default, there is no limit to the total number of jobs that can run on a host, so if this parameter is set to 0, a very large number of jobs might be dispatched to a host all at once. This can overload your system to the point that it will be unable to create any more processes. It is not recommended to set this parameter to 0.

JOB_ACCEPT_INTERVAL set at the queue level (`lsb.queues`) overrides **JOB_ACCEPT_INTERVAL** set at the cluster level (`lsb.params`).

Note:

The parameter `JOB_ACCEPT_INTERVAL` only applies when there are running jobs on a host. In other words, when there are no running jobs on a host, a new job can go right away to this host. When the first job runs and finishes earlier than the next job accept interval (before the interval expires), this job accept interval is ignored and a job is dispatched to the same host.

For example, job1 is dispatched to host A. If job1 run time is 10 minutes, and the job accept interval is 1, **mbd_sleep_time** is 60 seconds. Therefore, no second job will be dispatched within 60 seconds to host A. However, if job1 run time is 5 seconds on host A, then after job1 completes, the host is available. Therefore, **JOB_ACCEPT_INTERVAL** policy allows 1 job to be dispatched to Host A as soon as possible.

Default

Set to 0 at time of installation. If otherwise undefined, then set to 1.

JOB_ARRAY_EVENTS_COMBINE**Syntax**

JOB_ARRAY_EVENTS_COMBINE=Y | N

Description

If enabled (set to Y), `mbatchd` generates additional events for operations on array jobs or elements.

If disabled (set to N), `mbatchd` generates the same event types as single jobs for each job array. This means that for job arrays with large array sizes, `mbatchd` performance will suffer due to writing or reading a large amount of events.

Note: Whether enabled or disabled, **mbatchd** will not generate additional events for operations on single jobs.

The following table shows the event records that will be shown when **JOB_ARRAY_EVENTS_COMBINE** is enabled or disabled.

Event types	Event logging behaviour	
	Enabled (Y)	Disabled (N)
JOB_MODIFY2	<p>When modified jobs are array jobs or array elements, the job IDs are recorded in the format of array index ranges.</p> <p>When a single job is modified, the job ID is recorded in the existing fields.</p>	The job IDs are recorded in the existing fields as individual job IDs. Additional fields for modified jobs are not used.
JOB_SIGNAL JOB_STATUS JOB_CLEAN JOB_SWITCH JOB_FINISH JOB_FINISH2 JOB_STATUS2 JOB_MOVE	<p>When the logged jobs are array job or multiple array elements, mbatchd will merge multiple events into a single one if the other fields can share the same data.</p> <p>When logging a single job or an array element, the fields of array index ranges will not be used and marked as 0. The existing field of the array index is used to record the array index of the job.</p>	The fields of array index ranges will not be used and marked as 0. The existing field of the array index will be used to record the array index of the job.

Default

Set to N by default.

JOB_ATA_DIR**Syntax**

`JOB_ATA_DIR=directory`

Description

The shared directory in which mbatchd saves the attached data of messages posted with the **bpost** command.

Use **JOB_ATA_DIR** if you use **bpost** and **bread** to transfer large data files between jobs and want to avoid using space in **LSB_SHAREDDIR**. By default, the **bread** command reads attachment data from the **JOB_ATA_DIR** directory.

JOB_ATA_DIR should be shared by all hosts in the cluster, so that any potential LSF master host can reach it. Like **LSB_SHAREDDIR**, the directory should be owned and writable by the primary LSF administrator. The directory must have at least 1 MB of free space.

The attached data will be stored under the directory in the format:

```
JOB_ATA_DIR/timestamp.jobid.msgs/msg$msgindex
```

On UNIX, specify an absolute path. For example:

```
JOB_ATA_DIR=/opt/share/lsf_work
```

On Windows, specify a UNC path or a path with a drive letter. For example:

```
JOB_ATA_DIR=\\HostA\temp\lsf_work
```

or

```
JOB_ATTA_DIR=D:\temp\lsf_work
```

After adding **JOB_ATTA_DIR** to `lsb.params`, use **badadmin reconfig** to reconfigure your cluster.

Valid values

JOB_ATTA_DIR can be any valid UNIX or Windows path up to a maximum length of 256 characters.

Default

Not defined

If **JOB_ATTA_DIR** is not specified, job message attachments are saved in `LSB_SHAREDIR/info/`.

JOB_CWD_TTL

Syntax

```
JOB_CWD_TTL=hours
```

Description

Specifies the time-to-live for the current working directory (CWD) of a job. LSF cleans created CWD directories after a job finishes based on the TTL value. LSF deletes the CWD for the job if LSF created that directory for the job. The following options are available:

- 0 - sbatchd deletes CWD when all process related to the job finishes.
- 2147483647 - Never delete the CWD for a job.
- 1 to 2147483646 - Delete the CWD for a job after the timeout expires.

The system checks the directory list every 5 minutes with regards to cleaning and deletes only the last directory of the path to avoid conflicts when multiple jobs share some parent directories. TTL will be calculated after the post-exec script finishes. When LSF (sbatchd) starts, it checks the directory list file and deletes expired CWDs.

If the value for this parameter is not set in the application profile, LSF checks to see if it is set at the cluster-wide level in `lsb.params`. If neither is set, the default value is used.

Default

Not defined. The value of 2147483647 is used, meaning the CWD is not deleted.

JOB_DEP_LAST_SUB

Description

Used only with job dependency scheduling.

If set to 1, whenever dependency conditions use a job name that belongs to multiple jobs, LSF evaluates only the most recently submitted job.

Otherwise, all the jobs with the specified name must satisfy the dependency condition.

Running jobs are not affected when **JOB_DEP_LAST_SUB** is changed.

To reevaluate job dependencies after changing **JOB_DEP_LAST_SUB**, run **badadmin reconfig**.

Default

Set to 1 at time of installation for the DEFAULT and PARALLEL configuration templates. If otherwise undefined, then 0 (turned off).

JOB_DISTRIBUTE_ON_HOST

Syntax

JOB_DISTRIBUTE_ON_HOST=pack | balance | any

Description

For NUMA CPU and memory affinity scheduling. Specifies how LSF distributes tasks for different jobs. The parameter has the following values:

pack

LSF attempts to pack tasks as tightly as possible across jobs. Topology nodes with fewer available resources will be favored for task allocations.

JOB_DISTRIBUTE_ON_HOST is not the same as the `distribute` clause on the command-line affinity resource requirement. JOB_DISTRIBUTE_ON_HOST decides how to distribute tasks between jobs, rather than within a job.

Use `pack` to allow your application to use memory locality.

balance

LSF attempts to distribute tasks equally across hosts' topology, while considering the allocations of all jobs. Topology nodes with more available resources will be favored for task allocations.

any

LSF attempts no job task placement optimization. LSF chooses the first available processor units for task placement.

When JOB_DISTRIBUTE_ON_HOST is not defined, any is the default value.

Default

Not defined. JOB_DISTRIBUTE_ON_HOST=any is used.

JOB_EXIT_RATE_DURATION

Description

Defines how long LSF waits before checking the job exit rate for a host. Used in conjunction with **EXIT_RATE** in `lsb.hosts` for LSF host exception handling.

If the job exit rate is exceeded for the period specified by **JOB_EXIT_RATE_DURATION**, LSF invokes `LSF_SERVERDIR/eadmin` to trigger a host exception.

Tuning

Tip:

Tune **JOB_EXIT_RATE_DURATION** carefully. Shorter values may raise false alarms, longer values may not trigger exceptions frequently enough.

Example

```
JOB_EXIT_RATE_DURATION=10
```

Default

5 minutes

JOB_GROUP_CLEAN

Setting the **JOB_GROUP_CLEAN** parameter allows for the automatic deletion of empty job groups.

Syntax

JOB_GROUP_CLEAN=Y | N | all

Valid values

- If **JOB_GROUP_CLEAN** = N, implicitly created job groups are not automatically deleted (even if they are empty) unless they are deleted manually using **bgdel**
- If **JOB_GROUP_CLEAN** = all, implicitly created job groups that are empty may be deleted automatically. Job groups can only be deleted automatically if they have no explicitly created child job groups, and haven't been attached to an SLA.
- If **JOB_GROUP_CLEAN** = Y, behavior is like JOB_GROUP_CLEAN=all, but job groups will not be deleted automatically if they have associated limits (directly or in descendant job groups).

Default

N (Implicitly created job groups are not automatically deleted unless they are deleted manually with **bgdel**.)

JOB_INCLUDE_POSTPROC

Syntax

JOB_INCLUDE_POSTPROC=Y | N

Description

Specifies whether LSF includes the post-execution processing of the job as part of the job. When set to Y:

- Prevents a new job from starting on a host until post-execution processing is finished on that host
- **sbatchd** sends both job finish status (**DONE** or **EXIT**) and post-execution processing status (**POST_DONE** or **POST_ERR**) to **mbatchd** at the same time

In MultiCluster job forwarding model, the **JOB_INCLUDE_POSTPROC** value in the receiving cluster applies to the job.

MultiCluster job lease model, the **JOB_INCLUDE_POSTPROC** value applies to jobs running on remote leased hosts as if they were running on local hosts.

The variable **LSB_JOB_INCLUDE_POSTPROC** in the user environment overrides the value of **JOB_INCLUDE_POSTPROC** in an application profile in `lsb.applications`. **JOB_INCLUDE_POSTPROC** in an application profile in `lsb.applications` overrides the value of **JOB_INCLUDE_POSTPROC** in `lsb.params`.

For CPU and memory affinity jobs, if **JOB_INCLUDE_POSTPROC**=Y, LSF does not release affinity resources until post-execution processing has finished, since slots are still occupied by the job during post-execution processing.

For SGI cpusets, if **JOB_INCLUDE_POSTPROC**=Y, LSF does not release the cpuset until post-execution processing has finished, even though post-execution processes are not attached to the cpuset.

Default

N (Post-execution processing is not included as part of the job, and a new job can start on the execution host before post-execution processing finishes.)

JOB_INFO_EVENT_DUMP_INTERVAL

Syntax

JOB_INFO_EVENT_DUMP_INTERVAL=*minutes*

Description

This parameter controls how frequently the job info events file (jobinfo cache) is rewritten.

If the dump interval is too frequent, it means a greater load on I/O operations. If the dump interval is too infrequent, events replay will take longer to finish.

Valid values

Any positive integer between 1 and 2147483646.

Default

15 (minutes)

See also

JOB_INFO_MEMORY_CACHE_SIZE in `lsb.params` and **LSB_JOBINFO_DIR** in `lsf.conf`.

JOB_INFO_MEMORY_CACHE_SIZE

Syntax

JOB_INFO_MEMORY_CACHE_SIZE=*num_MB*

Description

This parameter configures how much memory to use for the jobinfo cache. The unit is MB.

The jobinfo cache can reduce the load on the work directory filer by caching job information such as the job's environment variables, command-line and eexec data in memory, in a compressed format. The jobinfo cache is part of mbatchd memory.

Note: Reducing the load on the jobinfo cache means a reduction in mbatchd I/O operations to achieve better mbatchd performance on the other side.

A value of 0 disables the jobinfo cache.

Valid values

Valid values: 0 - 2147483646.

Note: If enabled, a minimum value of 500 MB is recommended.

Default

1024 (1 GB)

See also

JOB_INFO_EVENT_DUMP_INTERVAL in `lsb.params` and **LSB_JOBINFO_DIR** in `lsf.conf`.

JOB_POSITION_CONTROL_BY_ADMIN

Syntax

JOB_POSITION_CONTROL_BY_ADMIN=Y | N

Description

Allows LSF administrators to control whether users can use **btop** and **bbot** to move jobs to the top and bottom of queues. When JOB_POSITION_CONTROL_BY_ADMIN=Y, only the LSF administrator (including any queue administrators) can use **bbot** and **btop** to move jobs within a queue.

Default

N

See also

bbot, **btop**

JOB_POSTPROC_TIMEOUT

Syntax

JOB_POSTPROC_TIMEOUT=*minutes*

Description

Specifies a timeout in minutes for job post-execution processing. The specified timeout must be greater than zero.

If post-execution processing takes longer than the timeout, **sbatchd** reports that post-execution has failed (POST_ERR status), and kills the entire process group of the job's post-execution processes on UNIX and Linux. On Windows, only the parent process of the post-execution command is killed when the timeout expires. The child processes of the post-execution command are not killed.

If **JOB_INCLUDE_POSTPROC=Y**, and **sbatchd** kills the post-execution processes because the timeout has been reached, the CPU time of the post-execution processing is set to 0, and the job's CPU time does not include the CPU time of post-execution processing.

JOB_POSTPROC_TIMEOUT defined in an application profile in `lsb.applications` overrides the value in `lsb.params`. **JOB_POSTPROC_TIMEOUT** cannot be defined in the user environment.

In the MultiCluster job forwarding model, the **JOB_POSTPROC_TIMEOUT** value in the receiving cluster applies to the job.

In the MultiCluster job lease model, the **JOB_POSTPROC_TIMEOUT** value applies to jobs running on remote leased hosts as if they were running on local hosts.

When running host-based post execution processing, set **JOB_POSTPROC_TIMEOUT** to a value that gives the process enough time to run.

Default

2147483647 (Unlimited; post-execution processing does not time out.)

JOB_PREPROC_TIMEOUT

Syntax

JOB_PREPROC_TIMEOUT=*minutes*

Description

Specify a timeout in minutes for job pre-execution processing. The specified timeout must be an integer greater than zero. If the job's pre-execution processing takes longer than the timeout, LSF kills the job's pre-execution processes, kills the job with a pre-defined exit value of 98, and then requeues the job to the head of the queue. However, if the number of pre-execution retries has reached the limit, LSF suspends the job with PSUSP status instead of requeuing it.

JOB_PREPROC_TIMEOUT defined in an application profile in `lsb.applications` overrides the value in `lsb.params`. **JOB_PREPROC_TIMEOUT** cannot be defined in the user environment.

On UNIX and Linux, **sbatchd** kills the entire process group of the job's pre-execution processes.

On Windows, only the parent process of the pre-execution command is killed when the timeout expires, the child processes of the pre-execution command are not killed.

In the MultiCluster job forwarding model, **JOB_PREPROC_TIMEOUT** and the number of pre-execution retries defined in the receiving cluster apply to the job. When the number of attempts reaches the limit, the job returns to submission cluster and is rescheduled.

In the MultiCluster job lease model, **JOB_PREPROC_TIMEOUT** and the number of pre-execution retries defined in the submission cluster apply to jobs running on remote leased hosts, as if they were running on local hosts.

Default

Not defined. Pre-execution processing does not time out. However, when running host-based pre-execution processing, you cannot use the infinite value or it may fail. You must configure a reasonable value.

JOB_PRIORITY_OVER_TIME**Syntax**

`JOB_PRIORITY_OVER_TIME=increment/interval`

Description

JOB_PRIORITY_OVER_TIME enables automatic job priority escalation when **MAX_USER_PRIORITY** is also defined.

Valid values

increment

Specifies the value used to increase job priority every *interval* minutes. Valid values are positive integers.

interval

Specifies the frequency, in minutes, to *increment* job priority. Valid values are positive integers.

Default

Not defined.

Example

`JOB_PRIORITY_OVER_TIME=3/20`

Specifies that every 20 minute *interval* *increment* to job priority of pending jobs by 3.

See also

MAX_USER_PRIORITY

JOB_RUNLIMIT_RATIO

Syntax

JOB_RUNLIMIT_RATIO=*integer* | 0

Description

Specifies a ratio between a job run limit and the runtime estimate specified by **bsub -We** or **bmod -We, -We+, -Wep**. The ratio does not apply to the **RUNTIME** parameter in `lsb.applications`.

This ratio can be set to 0 and no restrictions are applied to the runtime estimate.

JOB_RUNLIMIT_RATIO prevents abuse of the runtime estimate. The value of this parameter is the ratio of run limit divided by the runtime estimate.

By default, the ratio value is 0. Only administrators can set or change this ratio. If the ratio changes, it only applies to newly submitted jobs. The changed value does not retroactively reapply to already submitted jobs.

If the ratio value is greater than 0:

- If the users specify a runtime estimate only (**bsub -We**), the job-level run limit will automatically be set to $runtime_ratio * runtime_estimate$. Jobs running longer than this run limit are killed by LSF. If the job-level run limit is greater than the hard run limit in the queue, the job is rejected.
- If the users specify a runtime estimate (**-We**) and job run limit (**-W**) at job submission, and the run limit is greater than $runtime_ratio * runtime_estimate$, the job is rejected.
- If the users modify the run limit to be greater than $runtime_ratio$, they must increase the runtime estimate first (**bmod -We**). Then they can increase the default run limit.
- LSF remembers the run limit is set with **bsub -W** or convert from $runtime_ratio * runtime_estimate$. When users modify the run limit with **bmod -Wn**, the run limit is automatically be set to $runtime_ratio * runtime_estimate$. If the run limit is set from $runtime_ratio$, LSF rejects the run limit modification.
- If users modify the runtime estimate with **bmod -We** and the run limit is set by the user, the run limit is $MIN(new_estimate * new_ratio, run_limit)$. If the run limit is set by $runtime_ratio$, the run limit is set to $new_estimate * new_ratio$.
- If users modify the runtime estimate by using **bmod -Wen** and the run limit is set by the user, it is not changed. If the run limit is set by $runtime_ratio$, it is set to unlimited.

In MultiCluster job forwarding model, **JOB_RUNLIMIT_RATIO** values in both the sending and receiving clusters apply to the job. The run limit in the receiving cluster cannot be greater than the value of $runtime * JOB_RUNLIMIT_RATIO$ in the receiving cluster. Some examples:

- Run limit (for example with **bsub -We**) is 10, **JOB_RUNLIMIT_RATIO**=5 in the sending cluster, **JOB_RUNLIMIT_RATIO**=0 in the receiving cluster—run limit=50, and the job will run
- Run limit (for example with **bsub -We**) is 10, **JOB_RUNLIMIT_RATIO**=5 in the sending cluster, **JOB_RUNLIMIT_RATIO**=3 in the receiving cluster—run limit=50, and the job will pend
- Run limit (for example with **bsub -We**) is 10, **JOB_RUNLIMIT_RATIO**=5 in the sending cluster, **JOB_RUNLIMIT_RATIO**=6 in the receiving cluster—run limit=50, and the job will run
- Run limit (for example with **bsub -We**) is 10, **JOB_RUNLIMIT_RATIO**=0 in the sending cluster, **JOB_RUNLIMIT_RATIO**=5 in the receiving cluster—run limit=50, and the job will run

MultiCluster job lease model, the **JOB_RUNLIMIT_RATIO** value applies to jobs running on remote leased hosts as if they were running on local hosts.

Default

0

JOB_SCHEDULING_INTERVAL

Syntax

JOB_SCHEDULING_INTERVAL=*min_time* [ms]

Description

The minimum scheduling cycle time. If the **mbschd** daemon finishes one round of scheduling but does not reach this time, **mbschd** puts itself into sleep mode. Specify in seconds, or include the keyword ms to specify in milliseconds. If set to 0, there is no interval between subsequent sessions and **mbschd** starts the next scheduler session immediately after the current session.

At the beginning of each job scheduling cycle, the scheduling daemon **mbschd** takes a snapshot of the state of resources in the cluster and makes allocation decisions based on this snapshot. Since preparing the resource snapshot puts a load on **mbatchd**, the scheduling cycle might be short in a relatively idle cluster, which overloads **mbatchd** due to the frequent resource snapshots taken.

This parameter places a restriction on the length of the job scheduling cycle, which prevents **mbschd** from taking frequent resource snapshots and limits the load that this places on **mbatchd**.

Tip: Run the **badmin perfmon view** command to determine the average value, which you can use to determine a reasonable value to set for the maximum scheduling cycle time.

Valid values

Number of seconds or milliseconds greater than or equal to zero (0).

Default

Set at time of installation to 1 second. If otherwise undefined, then set to 5 seconds.

JOB_SPOOL_DIR

Syntax

JOB_SPOOL_DIR=*dir*

Description

Specifies the directory for buffering batch standard output and standard error for a job.

When **JOB_SPOOL_DIR** is defined, the standard output and standard error for the job is buffered in the specified directory.

Files are copied from the submission host to a temporary file in the directory specified by the **JOB_SPOOL_DIR** on the execution host. LSF removes these files when the job completes.

If **JOB_SPOOL_DIR** is not accessible or does not exist, files are spooled to the default directory `$HOME/.lsbatch`.

For **bsub -is** and **bsub -Zs**, **JOB_SPOOL_DIR** must be readable and writable by the job submission user, and it must be shared by the master host and the submission host. If the specified directory is not accessible or does not exist, and **JOB_SPOOL_DIR** is specified, **bsub -is** cannot write to the default directory `LSB_SHAREDIR/cluster_name/lsf_indir`, and **bsub -Zs** cannot write to the default directory `LSB_SHAREDIR/cluster_name/lsf_cmddir`, and the job will fail.

As LSF runs jobs, it creates temporary directories and files under **JOB_SPOOL_DIR**. By default, LSF removes these directories and files after the job is finished. See **bsub** for information about job submission options that specify the disposition of these files.

On UNIX, specify an absolute path. For example:

```
JOB_SPOOL_DIR=/home/share/lsf_spool
```

On Windows, specify a UNC path or a path with a drive letter. For example:

```
JOB_SPOOL_DIR=\\HostA\share\spooldir
```

or

```
JOB_SPOOL_DIR=D:\share\spooldir
```

In a mixed UNIX/Windows cluster, specify one path for the UNIX platform and one for the Windows platform. Separate the two paths by a pipe character (|):

```
JOB_SPOOL_DIR=/usr/share/lsf_spool | \\HostA\share\spooldir
```

Valid value

JOB_SPOOL_DIR can be any valid path.

The path can include the following dynamic patterns, which are case sensitive:

- %H - first execution host name
- %P - project name
- %U - user name
- %C - execution cluster name
- %JG - job group (if not specified, it will be ignored)

LSF creates subdirectories with appropriate permissions, based on the following rules:

JOB_SPOOL_DIR is not defined

Job spool directory is \$HOME/.lsbatch.

JOB_SPOOL_DIR=/full_path/%dynamic_patterns

- If %U is the first dynamic pattern, LSF creates a subdirectory before the %U with 777 permission. LSF creates subdirectories after %U (including %U) with 700 permission.
- If %U is not included in the path, or it is not the first dynamic pattern, LSF creates all subdirectories with 777 permission.

JOB_SPOOL_DIR=/full_path with no dynamic patterns

Job spool directory is /full_path. LSF does not create any subdirectories.

When creating subdirectories in the job spool directory, LSF ignores any dynamic patterns in the path that are not available.

Jobs submitted with -is and -Zs do not support JOB_SPOOL_DIR paths containing dynamic patterns.

The entire path including **JOB_SPOOL_DIR** can up to 4094 characters on UNIX and Linux or up to 255 characters for Windows. This maximum path length includes:

- All directory and file paths attached to the **JOB_SPOOL_DIR** path
- Temporary directories and files that the LSF system creates as jobs run.

The path you specify for **JOB_SPOOL_DIR** should be as short as possible to avoid exceeding this limit.

Note: The first path must be UNIX and second path must be Windows.

Default

Not defined

Batch job output (standard output and standard error) is sent to the .lsbatch directory on the execution host:

lsb.params

- On UNIX: \$HOME/.lsbatch
- On Windows: %windir%\lsbtmpuser_id\.lsbatch

If %HOME% is specified in the user environment, uses that directory instead of %windir% for spooled output.

JOB_SWITCH2_EVENT (OBSOLETE)

Syntax

JOB_SWITCH2_EVENT=Y|N

Description

Obsolete as of 10.1.

Specify Y to allow mbatchd to generate the **JOB_SWITCH2** event log when switching a job array to another queue. If this parameter is not enabled, mbatchd will generate the old **JOB_SWITCH** event instead. The **JOB_SWITCH** event is generated for the switch of each array element. If the job array is very large, many **JOB_SWITCH** events are generated, causing mbatchd to use large amounts of memory to replay all the **JOB_SWITCH** events. This causes performance problems when mbatchd starts up.

JOB_SWITCH2 logs the switching of the array to another queue as one event instead of logging each array element separately. **JOB_SWITCH2** has these advantages:

- Reduces memory usage of mbatchd when replaying **bswitch destination_queue job_ID**, where job_ID is the job ID of the job array on which to operate.
- Reduces the time for reading records from **lsb.events** when mbatchd starts up.
- Reduces the size of **lsb.events**.

Default

N

JOB_TERMINATE_INTERVAL

Syntax

JOB_TERMINATE_INTERVAL=*seconds*

Description

UNIX only.

Specifies the time interval in seconds between sending SIGINT, SIGTERM, and SIGKILL when terminating a job. When a job is terminated, the job is sent SIGINT, SIGTERM, and SIGKILL in sequence with a sleep time of **JOB_TERMINATE_INTERVAL** between sending the signals. This allows the job to clean up if necessary.

Default

10 (seconds)

KILL_JOBS_OVER_RUNLIMIT

Syntax

KILL_JOBS_OVER_RUNLIMIT=*interval[:wait_time]*

Description

Enables the **mbatchd** daemon to kill jobs that are running over the defined **RUNLIMIT** value for a long period of time.

- *interval* specifies the checking interval, which is how often the **mbatchd** daemon checks if there are any jobs that run over the defined **RUNLIMIT** value, in minutes.
- *wait_time* is optional, and specifies how long the **mbatchd** daemon waits, in minutes, after the jobs reach the defined **RUNLIMIT** before killing these jobs. The default value is 10 minutes.

During each checking interval, **mbatchd** checks if any job runs over the defined **RUNLIMIT** by a value greater than the wait time. If there are jobs over this value, **mbatchd** directly kills the job. For any jobs that are killed because of this parameter setting, LSF logs an additional kill reason message, which is shown in the **bjobs -l** or **bhist -l** output.

Normally, when a job reaches the defined **RUNLIMIT**, the **sbatchd** daemon kills the job. However, under some conditions, the job still shows RUN status in **mbatchd**. This parameter allows **mbatchd** to clean up or remove jobs that reached the **RUNLIMIT** value for a long period of time.

Valid Values

For *interval*, specify an integer that is greater than 30 minutes. For *wait_time*, specify an integer.

Default

Not defined. If the **KILL_JOBS_OVER_RUNLIMIT** parameter is defined but the *wait_time* value is not defined, the default value for *wait_time* is 10 minutes.

LOCAL_MAX_PREEEXEC_RETRY**Syntax**

LOCAL_MAX_PREEEXEC_RETRY=*integer*

Description

The maximum number of times to attempt the pre-execution command of a job on the local cluster.

When this limit is reached, the default behavior of the job is defined by the **LOCAL_MAX_PREEEXEC_RETRY_ACTION** parameter in `lsb.params`, `lsb.queues`, or `lsb.applications`.

Valid values

0 < LOCAL_MAX_PREEEXEC_RETRY < 2147483647

Default

2147483647 (Unlimited number of pre-execution retry times.)

See also

LOCAL_MAX_PREEEXEC_RETRY_ACTION in `lsb.params`, `lsb.queues`, and `lsb.applications`.

LOCAL_MAX_PREEEXEC_RETRY_ACTION**Syntax**

LOCAL_MAX_PREEEXEC_RETRY_ACTION=SUSPEND | EXIT

Description

The default behavior of a job when it reaches the maximum number of times to attempt its pre-execution command on the local cluster (**LOCAL_MAX_PREEEXEC_RETRY**).

- If set to **SUSPEND**, the job is suspended and its status is set to **PSUSP**.
- If set to **EXIT**, the job exits and its status is set to **EXIT**. The job exits with the same exit code as the last pre-execution fail exit code.

This parameter is configured cluster-wide (**lsb.params**), at the queue level (**lsb.queues**), and at the application level (**lsb.applications**). The action specified in **lsb.applications** overrides **lsb.queues**, and **lsb.queues** overrides the **lsb.params** configuration.

Default

SUSPEND

See also

LOCAL_MAX_PREEEXEC_RETRY in **lsb.params**, **lsb.queues**, and **lsb.applications**.

LSB_SYNC_HOST_STAT_LIM**Syntax**

LSB_SYNC_HOST_STAT_LIM=Y|y|N|n

Description

Improves the speed with which **mbatchd** obtains host status, and therefore the speed with which LSF reschedules rerunnable jobs: the sooner LSF knows that a host has become unavailable, the sooner LSF reschedules any rerunnable jobs executing on that host. Useful for a large cluster.

This parameter is enabled by default. It allows **mbatchd** to periodically obtain the host status from the master **LIM** and verifies the status by polling each **sbatchd** at an interval. It is recommended not to disable this parameter because it may then take longer to get status updates.

Default

Y

See also

MBD_SLEEP_TIME in **lsb.params**

LSB_MAX_PROBE_SBD in **lsf.conf**

MAX_ACCT_ARCHIVE_FILE**Syntax**

MAX_ACCT_ARCHIVE_FILE=*integer*

Description

Enables automatic deletion of archived LSF accounting log files and specifies the archive limit.

Compatibility

ACCT_ARCHIVE_SIZE or **ACCT_ARCHIVE_AGE** should also be defined.

Example

```
MAX_ACCT_ARCHIVE_FILE=10
```

LSF maintains the current `lsb.acct` and up to 10 archives. Every time the old `lsb.acct.9` becomes `lsb.acct.10`, the old `lsb.acct.10` gets deleted.

See also

- **ACCT_ARCHIVE_AGE** also enables automatic archiving
- **ACCT_ARCHIVE_SIZE** also enables automatic archiving
- **ACCT_ARCHIVE_TIME** also enables automatic archiving

Default

Not defined. No deletion of `lsb.acct.n` files.

MAX_CONCURRENT_QUERY

Syntax

```
MAX_CONCURRENT_QUERY=integer
```

Description

This parameter applies to all query commands and defines the maximum batch queries (including job queries) that **mbatchd** can handle.

MAX_CONCURRENT_QUERY controls the maximum number of concurrent query commands under the following conditions:

- **LSB_QUERY_PORT** is not defined
- **LSB_QUERY_PORT** is defined and **LSB_QUERY_ENH** is Y

If the specified threshold is reached, the query commands will retry.

If **LSB_QUERY_PORT** is defined and **LSB_QUERY_ENH** is N, **MAX_CONCURRENT_QUERY** controls two thresholds separately:

- Maximum number of concurrent job related query commands
- Maximum number of concurrent other query commands

If either of the specified thresholds are reached, the query commands will retry.

Valid values

1-1000

Default

Set to 100 at time of installation. If otherwise undefined, then unlimited.

MAX_EVENT_STREAM_FILE_NUMBER

Syntax

```
MAX_EVENT_STREAM_FILE_NUMBER=integer
```

Description

Determines the maximum number of different `lsb.stream.utc` files that `mbatchd` uses. When **MAX_EVENT_STREAM_FILE_NUMBER** is reached, every time the size of the `lsb.stream` file reaches **MAX_EVENT_STREAM_SIZE**, the oldest `lsb.stream` file is overwritten.

Default

10

MAX_EVENT_STREAM_SIZE

Syntax

`MAX_EVENT_STREAM_SIZE=integer`

Description

Determines the maximum size in MB of the `lsb.stream` file used by system performance analysis tools.

When the **MAX_EVENT_STREAM_SIZE** size is reached, LSF logs a special event **EVENT_END_OF_STREAM**, closes the stream and moves it to `lsb.stream.0` and a new stream is opened.

All applications that read the file once the event **EVENT_END_OF_STREAM** is logged should close the file and reopen it.

Recommended value

2000 MB

Default

1024 MB

MAX_INFO_DIRS

Syntax

`MAX_INFO_DIRS=num_subdirs`

Description

The number of subdirectories under the `LSB_SHAREDIR/cluster_name/logdir/info` directory.

When **MAX_INFO_DIRS** is enabled, `mbatchd` creates the specified number of subdirectories in the `info` directory. These subdirectories are given an integer as its name, starting with 0 for the first subdirectory.

Important:

If you are using local duplicate event logging, you must run **badmin mbdrestart** after changing **MAX_INFO_DIRS** for the changes to take effect.

Valid values

1-1024

Default

Set to 500 at time of installation for the `HIGH_THROUGHPUT` configuration template. If otherwise undefined, then 0 (no subdirectories under the `info` directory; `mbatchd` writes all jobfiles to the `info` directory).

Example

```
MAX_INFO_DIRS=10
```

mbatchd creates ten subdirectories from `LSB_SHAREDIR/cluster_name/logdir/info/0` to `LSB_SHAREDIR/cluster_name/logdir/info/9`.

MAX_JOB_ARRAY_SIZE**Syntax**

```
MAX_JOB_ARRAY_SIZE=integer
```

Description

Specifies the maximum number of jobs in a job array that can be created by a user for a single job submission. The maximum number of jobs in a job array cannot exceed this value.

A large job array allows a user to submit a large number of jobs to the system with a single job submission.

Valid values

Specify a positive integer between 1 and 2147483646

Default

Set to 10000 at time of installation for the HIGH_THROUGHPUT configuration template. If otherwise undefined, then 1000.

MAX_JOB_ATTACH_SIZE**Syntax**

```
MAX_JOB_ATTACH_SIZE=integer | 0
```

Specify any number less than 20000.

Description

Maximum attached data size, in KB, that can be transferred to a job.

Maximum size for data attached to a job with the **bpost** command. Useful if you use **bpost** and **bread** to transfer large data files between jobs and you want to limit the usage in the current working directory.

0 indicates that jobs cannot accept attached data files.

Default

2147483647 (Unlimited; LSF does not set a maximum size of job attachments.)

MAX_JOB_NUM**Syntax**

```
MAX_JOB_NUM=integer
```

Description

The maximum number of finished jobs whose events are to be stored in the `lsb.events` log file.

lsb.params

Once the limit is reached, **mbatchd** starts a new event log file. The old event log file is saved as `lsb.events.n`, with subsequent sequence number suffixes incremented by 1 each time a new log file is started. Event logging continues in the new `lsb.events` file.

Default

Set at time of installation to 10000 for the DEFAULT configuration template and 100000 for the HIGH_THROUGHPUT configuration template. If otherwise undefined, then 1000.

MAX_JOB_PREEMPT

Syntax

`MAX_JOB_PREEMPT=integer`

Description

The maximum number of times a job can be preempted. Applies to queue-based preemption only.

Valid values

$0 < \text{MAX_JOB_PREEMPT} < 2147483647$

Default

2147483647 (Unlimited number of preemption times.)

MAX_JOB_PREEMPT_RESET

Syntax

`MAX_JOB_PREEMPT_RESET=Y/N`

Description

If **MAX_JOB_PREEMPT_RESET=N**, the job preempted count for **MAX_JOB_PREEMPT** is not reset when a started job is requeued, migrated, or rerun.

Default

Y. Job preempted counter resets to 0 once a started job is requeued, migrated, or rerun.

MAX_JOB_REQUEUE

Syntax

`MAX_JOB_REQUEUE=integer`

Description

The maximum number of times to requeue a job automatically.

Valid values

$0 < \text{MAX_JOB_REQUEUE} < 2147483647$

Default

2147483647 (Unlimited number of requeue times.)

MAX_JOBID

Syntax

MAX_JOBID=*integer*

Description

The job ID limit. The job ID limit is the highest job ID that LSF will ever assign, and also the maximum number of jobs in the system.

By default, LSF assigns job IDs up to 6 digits. This means that no more than 999999 jobs can be in the system at once.

Specify any integer from 999999 to 2147483646 (for practical purposes, you can use any 10-digit integer less than this value).

You cannot lower the job ID limit, but you can raise it to 10 digits. This allows longer term job accounting and analysis, and means you can have more jobs in the system, and the job ID numbers will roll over less often.

LSF assigns job IDs in sequence. When the job ID limit is reached, the count rolls over, so the next job submitted gets job ID "1". If the original job 1 remains in the system, LSF skips that number and assigns job ID "2", or the next available job ID. If you have so many jobs in the system that the low job IDs are still in use when the maximum job ID is assigned, jobs with sequential numbers could have totally different submission times.

Example

```
MAX_JOBID=125000000
```

Default

999999

MAX_JOBINFO_QUERY_PERIOD

Syntax

MAX_JOBINFO_QUERY_PERIOD=*integer*

Description

Maximum time for job information query commands (for example, with **bjobs**) to wait.

When the time arrives, the query command processes exit, and all associated threads are terminated.

If the parameter is not defined, query command processes will wait for all threads to finish.

Specify a multiple of **MBD_REFRESH_TIME**.

Valid values

Any positive integer greater than or equal to one (1)

Default

2147483647 (Unlimited wait time.)

See also

LSB_BLOCK_JOBINFO_TIMEOUT in `lsf.conf`

MAX_PEND_JOBS

Syntax

MAX_PEND_JOBS=*integer*

Description

The maximum number of pending jobs in the cluster.

This is the hard system-wide pending job threshold. No user or user group can exceed this limit unless the job is forwarded from a remote cluster.

If the user or user group submitting the job has reached the pending job threshold as specified by **MAX_PEND_JOBS**, LSF will reject any further job submission requests sent by that user or user group. The system will continue to send the job submission requests with the interval specified by **SUB_TRY_INTERVAL** in `lsb.params` until it has made a number of attempts equal to the **LSB_NTRIES** environment variable. If **LSB_NTRIES** is not defined and LSF rejects the job submission request, the system will continue to send the job submission requests indefinitely as the default behavior.

Default

2147483647 (Unlimited number of pending jobs.)

See also

SUB_TRY_INTERVAL

MAX_PEND_SLOTS

MAX_PEND_SLOTS

Syntax

MAX_PEND_SLOTS=*integer*

Description

The maximum number of pending job slots available in the system.

This is the hard system-wide pending slots threshold. No user or user group can exceed this limit unless the job is forwarded from a remote cluster.

If the user or user group submitting the job has reached the pending job slots threshold as specified by **MAX_PEND_SLOTS**, LSF will reject any further job submission requests sent by that user or user group. The system will continue to send the job submission requests with the interval specified by **SUB_TRY_INTERVAL** in `lsb.params` until it has made a number of attempts equal to the **LSB_NTRIES** environment variable. If **LSB_NTRIES** is not defined and LSF rejects the job submission request, the system will continue to send the job submission requests indefinitely as the default behavior.

Default

2147483647 (Unlimited number of pending jobs.)

See also

SUB_TRY_INTERVAL

MAX_PEND_JOBS

MAX_PREEEXEC_RETRY

Syntax

MAX_PREEEXEC_RETRY=*integer*

Description

MultiCluster job forwarding model only. The maximum number of times to attempt the pre-execution command of a job from a remote cluster.

If the job's pre-execution command fails all attempts, the job is returned to the submission cluster.

Valid values

0 < MAX_PREEEXEC_RETRY < 2147483647

Default

5

MAX_PROTOCOL_INSTANCES

Syntax

MAX_PROTOCOL_INSTANCES=*integer*

Description

For LSF IBM Parallel Environment (PE) integration. Specify the number of parallel communication paths (windows) available to the protocol on each network. If number of windows specified for the job (with the `instances` option of `bsub -network` or the `NETWORK_REQ` parameter in `lsb.queues` or `lsb.applications`) is greater than the specified maximum value, LSF rejects the job.

Specify MAX_PROTOCOL_INSTANCES in a queue (`lsb.queues`) or cluster-wide in `lsb.params`. The value specified in a queue overrides the value specified in `lsb.params`.

LSF_PE_NETWORK_NUM must be defined to a non-zero value in `lsf.conf` for MAX_PROTOCOL_INSTANCES to take effect and for LSF to run PE jobs. If LSF_PE_NETWORK_NUM is not defined or is set to 0, the value of MAX_PROTOCOL_INSTANCES is ignored with a warning message.

Default

2

MAX_SBD_CONNS

Sets the maximum number of open file connections between `mbatchd` and `sbatchd`. The system sets `MAX_SBD_CONNS` automatically during `mbatchd` startup.

Syntax

MAX_SBD_CONNS=*integer*

Description

`MAX_SBD_CONNS` and `LSB_MAX_JOB_DISPATCH_PER_SESSION` affect the number of file descriptors. To decrease the load on the master LIM you should not configure the master host as the first host for the `LSF_SERVER_HOSTS` parameter.

The default values for `MAX_SBD_CONNS` and `LSB_MAX_JOB_DISPATCH_PER_SESSION` are set during `mbatchd` startup. They are not changed dynamically. If hosts are added dynamically, `mbatchd` does not

increase their values. Once all the hosts have been added, you must run **badmin mbdrestart** to set the correct values. If you know in advance that your cluster will dynamically grow or shrink, you should configure these parameters beforehand.

Default

MAX_SBD_CONNS = **numOfHosts** + (2 * **LSB_MAX_JOB_DISPATCH_PER_SESSION**)+200. This formula does not provide the exact number of **sbatchd** connections because it also calculates the lost and found hosts. Therefore, the calculated number of connections might be a few more than this theoretical number.

MAX_SBD_FAIL**Syntax**

MAX_SBD_FAIL=*integer*

Description

The maximum number of retries for reaching a non-responding slave batch daemon, **sbatchd**.

The minimum interval between retries is defined by **MBD_SLEEP_TIME**/10. If **mbatchd** fails to reach a host and has retried **MAX_SBD_FAIL** times, the host is considered unavailable or unreachable.

After **mbatchd** tries to reach a host **MAX_SBD_FAIL** number of times, **mbatchd** reports the host status as unavailable or unreachable.

When a host becomes unavailable, **mbatchd** assumes that all jobs running on that host have exited and that all rerunnable jobs (jobs submitted with the **bsub -r** option) are scheduled to be rerun on another host.

Default

3

MAX_TOTAL_TIME_PREEMPT**Syntax**

MAX_TOTAL_TIME_PREEMPT=*integer*

Description

The accumulated preemption time in minutes after which a job cannot be preempted again, where *minutes* is wall-clock time, not normalized time.

The parameter of the same name in **lsb.queues** overrides this parameter. The parameter of the same name in **lsb.applications** overrides both this parameter and the parameter of the same name in **lsb.queues**.

Valid values

Any positive integer greater than or equal to one (1)

Default

Unlimited

MAX_USER_PRIORITY

Syntax

MAX_USER_PRIORITY=*integer*

Description

Enables user-assigned job priority and specifies the maximum job priority a user can assign to a job. LSF and queue administrators can assign a job priority higher than the specified value for jobs they own.

Compatibility

User-assigned job priority changes the behavior of **btop** and **bbot**.

Example

```
MAX_USER_PRIORITY=100
```

Specifies that 100 is the maximum job priority that can be specified by a user.

Default

Not defined.

See also

- **bsub**, **bmod**, **btop**, **bbot**
- **JOB_PRIORITY_OVER_TIME**

MBD_EGO_CONNECT_TIMEOUT

Syntax

MBD_EGO_CONNECT_TIMEOUT=*seconds*

Description

For EGO-enabled SLA scheduling, timeout parameter for network I/O connection with EGO vemkd.

Default

0 seconds

MBD_EGO_READ_TIMEOUT

Syntax

MBD_EGO_READ_TIMEOUT=*seconds*

Description

For EGO-enabled SLA scheduling, timeout parameter for network I/O read from EGO vemkd after connection with EGO.

Default

0 seconds

MBD_EGO_TIME2LIVE

Syntax

MBD_EGO_TIME2LIVE=*minutes*

Description

For EGO-enabled SLA scheduling, specifies how long EGO should keep information about host allocations in case mbatchd restarts,

Default

0 minutes

MBD_QUERY_CPUS

Syntax

MBD_QUERY_CPUS=*cpu_list*

cpu_list defines the list of master host CPUS on which the mbatchd child query processes can run. Format the list as a white-space delimited list of CPU numbers.

For example, if you specify

```
MBD_QUERY_CPUS=1 2 3
```

the mbatchd child query processes will run only on CPU numbers 1, 2, and 3 on the master host.

Description

This parameter allows you to specify the master host CPUs on which mbatchd child query processes can run (hard CPU affinity). This improves mbatchd scheduling and dispatch performance by binding query processes to specific CPUs so that higher priority mbatchd processes can run more efficiently.

When you define this parameter, LSF runs mbatchd child query processes *only* on the specified CPUs. The operating system can assign other processes to run on the same CPU; however, if utilization of the bound CPU is lower than utilization of the unbound CPUs.

Important

1. You can specify CPU affinity only for master hosts that use one of the following operating systems:
 - Linux 2.6 or higher
 - Solaris 10 or higher
2. If failover to a master host candidate occurs, LSF maintains the hard CPU affinity, provided that the master host candidate has the same CPU configuration as the original master host. If the configuration differs, LSF ignores the CPU list and reverts to default behavior.

Related parameters

To improve scheduling and dispatch performance of all LSF daemons, you should use **MBD_QUERY_CPUS** together with **EGO_DAEMONS_CPUS** (in `ego.conf`), which controls LIM CPU allocation, and **LSF_DAEMONS_CPUS**, which binds mbatchd and **mbschd** daemon processes to specific CPUs so that higher priority daemon processes can run more efficiently. To get best performance, CPU allocation for all four daemons should be assigned their own CPUs. For example, on a 4 CPU SMP host, the following configuration will give the best performance:

```
EGO_DAEMONS_CPUS=0 LSF_DAEMONS_CPUS=1:2 MBD_QUERY_CPUS=3
```

Default

Not defined

See also

LSF_DAEMONS_CPUS in `lsf.conf`

MBD_REFRESH_TIME**Syntax**

`MBD_REFRESH_TIME=seconds [min_refresh_time]`

where *min_refresh_time* defines the minimum time (in seconds) that the child **mbatchd** will stay to handle queries.

Description

Time interval, in seconds, when **mbatchd** will fork a new child **mbatchd** to service query requests to keep information sent back to clients updated. A child **mbatchd** processes query requests creating threads.

To enable **MBD_REFRESH_TIME** you must specify **LSB_QUERY_PORT** in `lsf.conf`. The child **mbatchd** listens to the port number specified by **LSB_QUERY_PORT** and creates threads to service requests until the job changes status, a new job is submitted, or **MBD_REFRESH_TIME** has expired.

- If **MBD_REFRESH_TIME** is less than or equal to *min_refresh_time*, the child **mbatchd** exits at **MBD_REFRESH_TIME** even if the job changes status or a new job is submitted before **MBD_REFRESH_TIME** expires.
- If **MBD_REFRESH_TIME** is greater than *min_refresh_time*:
 - the child **mbatchd** exits at *min_refresh_time* if a job changes status or a new job is submitted before the *min_refresh_time*
 - the child **mbatchd** exits after the *min_refresh_time* when a job changes status or a new job is submitted
 - the child **mbatchd** exits at **MBD_REFRESH_TIME** if no job changes status or a new job is submitted

The value of this parameter must be between 0 and 300. Any values specified out of this range are ignored, and the system default value is applied.

The **bjobs** command may not display up-to-date information if two consecutive query commands are issued before a child **mbatchd** expires because child **mbatchd** job information is not updated. If you use the **bjobs** command and do not get up-to-date information, you may need to decrease the value of this parameter. Note, however, that the lower the value of this parameter, the more you negatively affect performance.

The number of concurrent requests is limited by the number of concurrent threads that a process can have. This number varies by platform:

- Sun Solaris, 2500 threads per process
- AIX, 512 threads per process
- HP-UX, 64 threads per process
- Linux does not have a direct threads per process limit, but instead has a limit on the total number of processes on the system. To determine this number, run the following command:

```
cat /proc/sys/kernel/threads-max
```

Set the **NEWJOB_REFRESH** parameter in the `lsb.params` as Y to enable a child **mbatchd** daemon to get up to date information about new jobs from the parent **mbatchd** daemon after the child **mbatchd** daemon was created. If **NEWJOB_REFRESH=Y** is enabled, set the **MBD_REFRESH** parameter to a value greater than 10 seconds.

Required parameters

LSB_QUERY_PORT must be enabled in `lsf.conf`.

Restrictions

Applies only to UNIX operating systems that support thread programming.

Valid Values

5-300 seconds

Default

The default value for the minimum refresh time is adjusted automatically based on the number of jobs in the system:

- If there are less than 500,000 jobs in the system, the default value is 10 seconds.
- If there are more than 500,000 jobs in the system, the default value is 10 seconds + $(\#jobs - 500,000) / 100,000$.

See also

LSB_QUERY_PORT in `lsf.conf`, **NEWJOB_REFRESH** in `lsb.params`

MBD_SLEEP_TIME

Syntax

`MBD_SLEEP_TIME=seconds`

Description

Used in conjunction with the parameters **SLOT_RESERVE**, **MAX_SBD_FAIL**, and **JOB_ACCEPT_INTERVAL**. Amount of time in seconds used for calculating parameter values.

Default

Set at installation to 10 seconds. If not defined, 60 seconds.

MBD_USE_EGO_MXJ

Syntax

`MBD_USE_EGO_MXJ=Y | N`

Description

By default, when EGO-enabled SLA scheduling is configured, EGO allocates an entire host to LSF, which uses its own MXJ definition to determine how many slots are available on the host. LSF gets its host allocation from EGO, and runs as many jobs as the LSF configured MXJ for that host dictates.

MBD_USE_EGO_MXJ forces LSF to use the job slot maximum configured in the EGO consumer. This allows partial sharing of hosts (for example, a large SMP computer) among different consumers or workload managers. When **MBD_USE_EGO_MXJ** is set, LSF schedules jobs based on the number of slots allocated from EGO. For example, if `hostA` has 4 processors, but EGO allocates 2 slots to an EGO-enabled SLA consumer. LSF can schedule a maximum of 2 jobs from that SLA on `hostA`.

Default

N (mbatcthd uses the LSF MXJ)

MC_PENDING_REASON_PKG_SIZE**Syntax**

MC_PENDING_REASON_PKG_SIZE=*kilobytes* | 0

Description

MultiCluster job forwarding model only. Pending reason update package size, in KB. Defines the maximum amount of pending reason data this cluster will send to submission clusters in one cycle.

Specify the keyword 0 (zero) to disable the limit and allow any amount of data in one package.

Default

512

MC_PENDING_REASON_UPDATE_INTERVAL**Syntax**

MC_PENDING_REASON_UPDATE_INTERVAL=*seconds* | 0

Description

MultiCluster job forwarding model only. Pending reason update interval, in seconds. Defines how often this cluster will update submission clusters about the status of pending MultiCluster jobs.

Specify the keyword 0 (zero) to disable pending reason updating between clusters.

Default

300

MC_PLUGIN_SCHEDULE_ENHANCE**Syntax**

MC_PLUGIN_SCHEDULE_ENHANCE=RESOURCE_ONLY

MC_PLUGIN_SCHEDULE_ENHANCE=COUNT_PREEMPTABLE [HIGH_QUEUE_PRIORITY]

[PREEMPTABLE_QUEUE_PRIORITY] [PENDING_WHEN_NOSLOTS]

MC_PLUGIN_SCHEDULE_ENHANCE=DYN_CLUSTER_WEIGHTING

Note:

When any one of **HIGH_QUEUE_PRIORITY**, **PREEMPTABLE_QUEUE_PRIORITY** or **PENDING_WHEN_NOSLOTS** is defined, **COUNT_PREEMPTABLE** is enabled automatically.

Description

MultiCluster job forwarding model only. The parameter **MC_PLUGIN_SCHEDULE_ENHANCE** enhances the scheduler for the MultiCluster job forwarding model based on the settings selected. Use in conjunction with **MC_PLUGIN_UPDATE_INTERVAL** to set the data update interval between remote clusters.

MC_PLUGIN_UPDATE_INTERVAL must be a non-zero value to enable the MultiCluster enhanced scheduler.

With the parameter **MC_PLUGIN_SCHEDULE_ENHANCE** set to a valid value, remote resources are considered as if **MC_PLUGIN_REMOTE_RESOURCE=Y** regardless of the actual setting. In addition the submission cluster scheduler considers specific execution queue resources when scheduling jobs.

Note:

The parameter **MC_PLUGIN_SCHEDULE_ENHANCE** was introduced in LSF Version 7 Update 6. All clusters within a MultiCluster configuration must be running a version of LSF containing this parameter to enable the enhanced scheduler.

After a MultiCluster connection is established, counters take the time set in **MC_PLUGIN_UPDATE_INTERVAL** to update. Scheduling decisions made before this first interval has passed do not accurately account for remote queue workload.

Default

Not defined.

The enhanced MultiCluster scheduler is not used. If **MC_PLUGIN_REMOTE_RESOURCE=Y** in `lsf.conf` remote resource availability is considered before jobs are forwarded to the queue with the most available slots.

See also

MC_PLUGIN_UPDATE_INTERVAL in `lsb.params`.

MC_PLUGIN_REMOTE_RESOURCE in `lsf.conf`.

MC_PLUGIN_UPDATE_INTERVAL

Syntax

`MC_PLUGIN_UPDATE_INTERVAL=seconds | 0`

Description

MultiCluster job forwarding model only; set for the execution cluster. The number of seconds between data updates between clusters.

A non-zero value enables collection of remote cluster queue data for use by the submission cluster enhanced scheduler.

Suggested value when enabled is **MBD_SLEEP_TIME** (default is 20 seconds).

A value of 0 disables collection of remote cluster queue data.

Default

0

See Also

MC_PLUGIN_SCHEDULE_ENHANCE in `lsf.params`.

MC_RECLAIM_DELAY

Syntax

`MC_RECLAIM_DELAY=minutes`

Description

MultiCluster resource leasing model only. The reclaim interval (how often to reconfigure shared leases) in minutes.

Shared leases are defined by Type=shared in the lsb.resources HostExport section.

Default

10 (minutes)

MC_RESOURCE_MATCHING_CRITERIA**Syntax**

MC_RESOURCE_MATCHING_CRITERIA=<rc1> <rc2>...

Description

This parameter is configured on the MultiCluster execution side and defines numeric and string resources for the execution cluster to pass back to the submission cluster. The execution cluster makes the submission cluster aware of what resources and their values are listed so that the submission cluster can make better forwarding decisions.

You can define resources that meet the following criteria:

- User defined numeric and string resources
- Host based resources, for example:
 - Resources defined in the RESOURCE column in the Host section of the lsf.cluster file.
 - Resource location as [default] or value@[default] in the Resource Map section of the lsf.cluster file.
- Non-consumable resources.

Although you can configure dynamic resources as criterion, they should be as close to static as possible to make forwarding decisions accurately. The number of criterion and values for each resource should be limited within a reasonable range to prevent deterioration of forward scheduling performance.

The behavior for **MC_PLUGIN_REMOTE_RESOURCE** is the default behavior and is kept for compatibility.

Default

None

MC_RUSAGE_UPDATE_INTERVAL**Syntax**

MC_RUSAGE_UPDATE_INTERVAL=*seconds*

Description

MultiCluster only. Enables resource use updating for MultiCluster jobs running on hosts in the cluster and specifies how often to send updated information to the submission or consumer cluster.

Default

300

MC_SORT_BY_SUBMIT_TIME

Syntax

MC_SORT_BY_SUBMIT_TIME=Y|y|N|n

Description

Enabling this parameter in a MultiCluster environment allows forwarded jobs on the execution cluster to be sorted and run based on their original submission time (instead of their forwarded time).

When the maximum rescheduled time has been reached and the pending jobs are rescheduled on the execution cluster, they are ordered based on their original submission time (the time when the job was first submitted on the submission cluster) and not the forwarding time (the time when the job was re-forwarded to the execution cluster).

Jobs forwarded to the execution cluster using **brequeue -a** or **brequeue -p** are also sorted based on the submission time.

Note : This parameter does not change the behaviour of **bswitch**, **bbot** or **btop**.

- After **bswitch**, LSF forwards the job based on job switch time instead of the submission time.
- Using **btop** or **bbot** on the submission cluster does not affect the position of the jobs at the execution cluster or future forwarding.
- User can **btop** the job at the execution cluster but after the job is recalled and forwarded again, LSF will order the job based on the original submission time and the previous **btop** position is gone.

Default

N

MIN_SWITCH_PERIOD

Syntax

MIN_SWITCH_PERIOD=*seconds*

Description

The minimum period in seconds between event log switches.

Works together with **MAX_JOB_NUM** to control how frequently **mbatchd** switches the file. **mbatchd** checks if **MAX_JOB_NUM** has been reached every **MIN_SWITCH_PERIOD** seconds. If **mbatchd** finds that **MAX_JOB_NUM** has been reached, it switches the events file.

To significantly improve the performance of **mbatchd** for large clusters, set this parameter to a value equal to or greater than 600. This causes **mbatchd** to fork a child process that handles event switching, thereby reducing the load on **mbatchd**. **mbatchd** terminates the child process and appends delta events to new events after the **MIN_SWITCH_PERIOD** has elapsed.

Default

Set to 1800 at time of installation for the HIGH_THROUGHPUT configuration template. If otherwise undefined, then 0 (no minimum period, log switch frequency is not restricted).

See also

MAX_JOB_NUM

NEWJOB_REFRESH

Syntax

NEWJOB_REFRESH=Y | N

Description

Enables a child **mbatchd** to get up to date information about new jobs from the parent **mbatchd**. When set to Y, job queries with **bjobs** display new jobs submitted after the child **mbatchd** was created.

If you have enabled multithreaded **mbatchd** support, the **bjobs** command may not display up-to-date information if two consecutive query commands are issued before a child **mbatchd** expires because child **mbatchd** job information is not updated. Use NEWJOB_REFRESH=Y to enable the parent **mbatchd** to push new job information to a child **mbatchd**

When NEWJOB_REFRESH=Y, as users submit new jobs, the parent **mbatchd** pushes the new job event to the child **mbatchd**. The parent **mbatchd** transfers the following kinds of new jobs to the child **mbatchd**:

- Newly submitted jobs
- Restarted jobs
- Remote lease model jobs from the submission cluster
- Remote forwarded jobs from the submission cluster

When NEWJOB_REFRESH=Y, you should set **MBD_REFRESH_TIME** to a value greater than 10 seconds.

Required parameters

LSB_QUERY_PORT must be enabled in `lsf.conf`.

Restrictions

Applies only to UNIX operating systems that support thread programming.

The parent **mbatchd** only pushes the new job event to a child **mbatchd**. The child **mbatchd** is not aware of status changes of existing jobs. The child **mbatchd** will not reflect the results of job control commands (**bmod**, **bmig**, **bswitch**, **btot**, **bbot**, **brequeue**, **bstop**, **brresume**, and so on) invoked after the child **mbatchd** is created.

Default

Set to Y at time of installation for the DEFAULT and PARALLEL configuration templates. If otherwise undefined, then N (new jobs are not pushed to the child **mbatchd**).

See also

MBD_REFRESH_TIME

NO_PREEMPT_FINISH_TIME

Syntax

NO_PREEMPT_FINISH_TIME=*minutes* | *percentage*

Description

Prevents preemption of jobs that will finish within the specified number of minutes or the specified percentage of the estimated run time or run limit.

Specifies that jobs due to finish within the specified number of minutes or percentage of job duration should not be preempted, where *minutes* is wall-clock time, not normalized time. Percentage must be greater than 0 or less than 100% (between 1% and 99%).

For example, if the job run limit is 60 minutes and `NO_PREEMPT_FINISH_TIME=10%`, the job cannot be preempted after it running 54 minutes or longer.

If you specify percentage for **NO_PREEMPT_RUN_TIME**, requires a run time (**bsub -We** or **RUNTIME** in `lsb.applications`), or run limit to be specified for the job (**bsub -W**, or **RUNLIMIT** in `lsb.queues`, or **RUNLIMIT** in `lsb.applications`)

Default

Not defined.

NO_PREEMPT_INTERVAL

Specifies the number of minutes a preemptable job can run before it is preempted. If the uninterrupted run time of a preemptable job is longer than the specified time, it can be preempted.

Syntax

`NO_PREEMPT_INTERVAL=minutes`

The value of *minutes* is wall-clock time, not normalized time.

Description

The `NO_PREEMPT_INTERVAL=0` parameter allows immediate preemption of jobs as soon as they start or resume running.

For example, if a job A needs to preempt other candidate preemptable jobs B, C, and D, the **NO_PREEMPT_INTERVAL** parameter determines which job is preempted:

- Run time of job B and job C is less than the **NO_PREEMPT_INTERVAL** parameter: job B and C are not preempted.
- Run time of job D is greater than or equal to the **NO_PREEMPT_INTERVAL** parameter: job D is preempted.

The parameter of the same name in the `lsb.queues` file overrides this parameter. The parameter of the same name in the `lsb.applications` file overrides both this parameter and the parameter of the same name in the `lsb.queues` file.

Default

0

NO_PREEMPT_RUN_TIME

Syntax

`NO_PREEMPT_RUN_TIME=minutes | percentage`

Description

Prevents preemption of jobs that have been running for the specified number of minutes or the specified percentage of the estimated run time or run limit.

Specifies that jobs that have been running for the specified number of minutes or longer should not be preempted, where *minutes* is wall-clock time, not normalized time. Percentage must be greater than 0 or less than 100% (between 1% and 99%).

For example, if the job run limit is 60 minutes and `NO_PREEMPT_RUN_TIME=50%`, the job cannot be preempted after it running 30 minutes or longer.

If you specify percentage for **NO_PREEMPT_RUN_TIME**, requires a run time (**bsub -We** or **RUNTIME** in `lsb.applications`), or run limit to be specified for the job (**bsub -W**, or **RUNLIMIT** in `lsb.queues`, or **RUNLIMIT** in `lsb.applications`)

Default

Not defined.

MAX_JOB_MSG_NUM**Syntax**

MAX_JOB_MSG_NUM=*integer* | 0

Description

Maximum number of message slots for each job. Maximum number of messages that can be posted to a job with the **bpost** command.

0 indicates that jobs cannot accept external messages.

Default

128

ORPHAN_JOB_TERM_GRACE_PERIOD**Syntax**

ORPHAN_JOB_TERM_GRACE_PERIOD=*seconds*

Description

If defined, enables automatic orphan job termination at the cluster level which applies to all dependent jobs; otherwise it is disabled. This parameter is also used to define a cluster-wide termination grace period to tell LSF how long to wait before killing orphan jobs. Once configured, automatic orphan job termination applies to all dependent jobs in the cluster.

- **ORPHAN_JOB_TERM_GRACE_PERIOD = 0**: Automatic orphan job termination is enabled in the cluster but no termination grace period is defined. A dependent job can be terminated as soon as it is found to be an orphan.
- **ORPHAN_JOB_TERM_GRACE_PERIOD > 0**: Automatic orphan job termination is enabled and the termination grace period is set to the specified number of seconds. This is the minimum time LSF will wait before terminating an orphan job. In a multi-level job dependency tree, the grace period is not repeated at each level, and all direct and indirect orphans of the parent job can be terminated by LSF automatically after the grace period has expired.

The valid range of values is any integer greater than or equal to 0 and less than 2147483647.

Default

Not defined. Automatic orphan termination is disabled.

PARALLEL_SCHED_BY_SLOT**Syntax**

PARALLEL_SCHED_BY_SLOT=*y* | Y

Description

If defined, LSF schedules jobs based on the number of slots assigned to the hosts instead of the number of CPUs. For example, if MXJ is set to "-", then LSF considers the default value for number of CPUs for that host. These slots can be defined by host in `lsb.hosts` or by slot limit in `lsb.resources`.

All slot-related messages still show the word “processors”, but actually refer to “slots” instead. Similarly, all scheduling activities also use slots instead of processors.

Default

Set to Y at time of installation. If otherwise undefined, then N (disabled).

See also

- JL/U and MXJ in `lsb.hosts`
- SLOTS and SLOTS_PER_PROCESSOR in `lsb.resources`

PEND_REASON_UPDATE_INTERVAL

Syntax

PEND_REASON_UPDATE_INTERVAL=*seconds*

Description

Time interval that defines how often pending reasons are calculated by the scheduling daemon `mbschd`.

Valid values

Any positive integer between 1 and 2147483646.

Default

Set to 60 seconds at time of installation for the HIGH_THROUGHPUT configuration template. If otherwise undefined, then 30 seconds.

PERFORMANCE_THRESHOLD_FILE

Syntax

PERFORMANCE_THRESHOLD_FILE=*full_file_path*

Description

Specifies the location of the performance threshold file for the cluster. This file contains the cluster-level threshold values for the minimize energy and minimize time policies, used with the energy aware scheduling automatic select CPU frequency feature.

Default

`LSF_ENVDIR/lsbatch/cluster_name/configdir/lsb.threshold`

PERSIST_LIVE_CONFIG

Syntax

PERSIST_LIVE_CONFIG=Y|y|N|n

Description

Specifies whether LSF persists live configuration changes so that they take effect after the `mbatchd` daemon restarts.

If PERSIST_LIVE_CONFIG=Y, LSF persists all live configuration changes so that they take effect after `mbatchd` restarts.

If `PERSIST_LIVE_CONFIG=N`, LSF does not persist live configuration changes and they do not take effect after restarting or reconfiguring `mbatchd`.

Default

Y

PG_SUSP_IT

Syntax

`PG_SUSP_IT=seconds`

Description

The time interval that a host should be interactively idle (it > 0) before jobs suspended because of a threshold on the pg load index can be resumed.

This parameter is used to prevent the case in which a batch job is suspended and resumed too often as it raises the paging rate while running and lowers it while suspended. If you are not concerned with the interference with interactive jobs caused by paging, the value of this parameter may be set to 0.

Default

180 seconds

PLAN

For use when the `ALLOCATION_PLANNER` parameter is enabled. Used to identify the jobs that are candidates for planning.

Syntax

`PLAN = Y | N | "<key>[value] ..."`

Description

LSF requires that the `ALLOCATION_PLANNER` parameter is enabled in order to use `PLAN=Y`.

Also defined at the application and queue levels. The precedence is: application, queue, global. For example, application level setting overrides the queue level setting.

The following key-value pairs are supported:

key	value	Default	Description
DELAY	positive integer	-	Number of minutes to delay before considering making a plan for a job following the job's submission time.
MAX_JOBS	positive integer	-	Maximum number of jobs that can have a plan concurrently.

Note:

The `PLAN` parameter replaces the existing `SLOT_RESERVE` parameter and `RESOURCE_RESERVE` parameter when the `ALLOCATION_PLANNER` parameter is enabled.

Default

N

POWER_ON_WAIT

Syntax

POWER_ON_WAIT=*time_seconds*

Description

Configures a wait time (in seconds) after a host is resumed and enters ok status, before dispatching a job. This is to allow other services on the host to restart and enter a ready state. The default value is 0 and is applied globally.

Default

0

POWER_RESET_CMD

Syntax

POWER_RESET_CMD=*command*

Description

Defines the reset operation script that will be called when handling a power reset request.

To allow the command to parse all its arguments as a host list, LSF uses the command in the format:

```
command host [host ...]
```

To show each host with its execution result (success (0) or fail (1)), the return value of the command follows the format:

```
host 0  
host 1  
...
```

Default

Not defined.

POWER_RESUME_CMD

Syntax

POWER_RESUME_CMD=command

Description

Defines the resume operation script that will be called when handling a resume request. An opposite operation to POWER_SUSPEND_CMD.

To allow the command to parse all its arguments as a host list, LSF uses the command in the format:

```
command host [host ...]
```

To show each host with its execution result (success (0) or fail (1)), the return value of the command follows the format:

```
host 0
host 1
...
```

Default

Not defined.

POWER_STATUS_LOG_MAX**Syntax**

POWER_STATUS_LOG_MAX=*number*

Description

Configures a trigger value for events switching. The default value is 10000. This value takes effect only if PowerPolicy (in lsb.resources) is enabled.

If a finished job number is not larger than the value of **MAX_JOB_NUM**, the event switch can also be triggered by POWER_STATUS_LOG_MAX, which works with MIN_SWITCH_PERIOD.

POWER_STATUS_LOG_MAX is not available with LSF Express edition.

Default

10000

POWER_SUSPEND_CMD**Syntax**

POWER_SUSPEND_CMD=*command*

Description

Defines the suspend operation script that will be called when handling a suspend request.

To allow the command to parse all its arguments as a host list, LSF uses the command in the format:

```
command host [host ...]
```

To show each host with its execution result (success (0) or fail (1)), the return value of the command follows the format:

```
host 0
host 1
...
```

Default

Not defined.

POWER_SUSPEND_TIMEOUT**Syntax**

POWER_SUSPEND_TIMEOUT=*integer*

Description

Defines the timeout value (in seconds) for power suspend, resume, and reset actions. When a power operation is not successful (for example, sbatchd does not reconnect when resuming a host) within the specified number of seconds, the action will be considered failed.

Default

600

PREEMPT_DELAY**Syntax**

PREEMPT_DELAY=*seconds*

Description

Preemptive jobs will wait the specified number of seconds from the submission time before preempting any low priority preemptable jobs. During the grace period, preemption will not be triggered, but the job can be scheduled and dispatched by other scheduling policies.

This feature can provide flexibility to tune the system to reduce the number of preemptions. It is useful to get better performance and job throughput. When the low priority jobs are short, if high priority jobs can wait a while for the low priority jobs to finish, preemption can be avoided and cluster performance is improved. If the job is still pending after the grace period has expired, the preemption will be triggered.

The waiting time is for preemptive jobs in the pending status only. It will not impact the preemptive jobs that are suspended.

The time is counted from the submission time of the jobs. The submission time means the time **mbatchd** accepts a job, which includes newly submitted jobs, restarted jobs (by **brestart**) or forwarded jobs from a remote cluster.

When the preemptive job is waiting, the pending reason is:

The preemptive job is allowing a grace period before preemption.

If you use an older version of **bjobs**, the pending reason is:

Unknown pending reason code <6701>;

The parameter is defined in `lsb.params`, `lsb.queues` (overrides `lsb.params`), and `lsb.applications` (overrides both `lsb.params` and `lsb.queues`).

Run **badmin reconfig** to make your changes take effect.

Default

Not defined (if the parameter is not defined anywhere, preemption is immediate).

PREEMPT_FOR**Syntax**

PREEMPT_FOR=[GROUP_JLP] [GROUP_MAX] [HOST_JLU] [LEAST_RUN_TIME] [MINI_JOB] [USER_JLP] [OPTIMAL_MINI_JOB] [BUSY]

Description

If preemptive scheduling is enabled, this parameter is used to disregard suspended jobs when determining if a job slot limit is exceeded, to preempt jobs with the shortest running time, and to optimize preemption of parallel jobs.

If preemptive scheduling is enabled, more lower-priority parallel jobs may be preempted than necessary to start a high-priority parallel job. Both running and suspended jobs are counted when calculating the number of job slots in use, except for the following limits:

- The total job slot limit for hosts, specified at the host level
- Total job slot limit for individual users, specified at the user level—by default, suspended jobs still count against the limit for user groups

Specify one or more of the following keywords. Use spaces to separate multiple keywords.

BUSY

Enables preemptive scheduling to preempt slots on the `closed_busy` host for all preemptive jobs except preemptive suspended jobs and only one job can preempt a `closed_busy` host at a time.

If the job cannot run in two minutes after making a reservation on the `closed_busy` host, the job preempts necessary slots on other candidate hosts or all preemptable slots on this `closed_busy` host.

After two minutes, if the `closed_busy` host is still the candidate host and there are no preemptable running jobs, this job will pend without any reservations.

If the job cannot run in five minutes after making a reservation on the `closed_busy` host, the job no longer considers this `closed_busy` host as a candidate host.

Tip: To decrease load on the `closed_busy` host for running preemptive pending jobs, specify `JOB_CONTROLS=SUSPEND[brqueuee $LSB_JOBID]` on preemptable queues.

To use the `closed_busy` host when submitting a higher priority preemptive job with the `select` statement, avoid removing hosts from consideration based on load indices

GROUP_JLP

Counts only running jobs when evaluating if a user group is approaching its per-processor job slot limit (`SLOTS_PER_PROCESSOR`, `USERS`, and `PER_HOST=all` in the `lsb.resources` file). Suspended jobs are ignored when this keyword is used.

GROUP_MAX

Counts only running jobs when evaluating if a user group is approaching its total job slot limit (`SLOTS`, `PER_USER=all`, and `HOSTS` in the `lsb.resources` file). Suspended jobs are ignored when this keyword is used. When preemptive scheduling is enabled, suspended jobs never count against the total job slot limit for individual users.

HOST_JLU

Counts only running jobs when evaluating if a user or user group is approaching its per-host job slot limit (`SLOTS` and `USERS` in the `lsb.resources` file). Suspended jobs are ignored when this keyword is used.

LEAST_RUN_TIME

Preempts the job that has been running for the shortest time. Run time is wall-clock time, not normalized run time.

MINI_JOB

Optimizes the preemption of parallel jobs by preempting only enough parallel jobs to start the high-priority parallel job.

OPTIMAL_MINI_JOB

Optimizes preemption of parallel jobs by preempting only low-priority parallel jobs based on the least number of jobs that will be suspended to allow the high-priority parallel job to start.

User limits and user group limits can interfere with preemption optimization of **OPTIMAL_MINI_JOB**. You should not configure **OPTIMAL_MINI_JOB** if you have user or user group limits configured.

You should configure `PARALLEL_SCHED_BY_SLOT=Y` when using **OPTIMAL_MINI_JOB**.

USER_JLP

Counts only running jobs when evaluating if a user is approaching their per-processor job slot limit (SLOTS_PER_PROCESSOR, USERS, and PER_HOST=all in the `lsb.resources` file). Suspended jobs are ignored when this keyword is used. Ignores suspended jobs when calculating the user-processor job slot limit for individual users. When preemptive scheduling is enabled, suspended jobs never count against the total job slot limit for individual users.

Default

0 (The parameter is not defined.)

Both running and suspended jobs are included in job slot limit calculations, except for job slots limits for hosts and individual users where only running jobs are ever included.

PREEMPT_JOBTYPE**Syntax**

PREEMPT_JOBTYPE=[EXCLUSIVE] [BACKFILL]

Description

If preemptive scheduling is enabled, this parameter enables preemption of exclusive and backfill jobs. Specify one or both of the following keywords. Separate keywords with a space.

EXCLUSIVE

Enables preemption of and preemption by exclusive jobs. **LSB_DISABLE_LIMLOCK_EXCL=Y** in `lsf.conf` must also be defined.

BACKFILL

Enables preemption of backfill jobs. Jobs from higher priority queues can preempt jobs from backfill queues that are either backfilling reserved job slots or running as normal jobs.

AFFINITY

Enables affinity resource preemption. Affinity resources (thread, core, socket, and NUMA) held by a suspended job can be used by a pending job through queue-based preemption, or through LSF License Scheduler preemption.

Default

Not defined. Exclusive and backfill jobs are only preempted if the exclusive low priority job is running on a different host than the one used by the preemptive high priority job.

PREEMPTABLE_RESOURCES**Syntax**

PREEMPTABLE_RESOURCES= res1 [res2] [res3]

Description

Enables preemption for resources (in addition to slots) when preemptive scheduling is enabled (has no effect if queue preemption is not enabled) and specifies the resources that will be preemptable. Specify shared resources (static or dynamic) that are numeric, decreasing, and releasable. One of the resources can be built-in resource mem, meaning that `res1` is also option if memory comes later in the list.

The default preemption action is the suspend the job. To force a job to release resources instead of suspending them, set `TERMINATE_WHEN=PREEMPT` in `lsb.queues`, or set **JOB_CONTROLS** in

lsb.queues and specify brequeue as the SUSPEND action. Some applications will release resources when sent the SIGTSTP signal. Use **JOB_CONTROLS** to send this signal to suspend the job.

To enable memory preemption, include mem in the **PREEMPTABLE_RESOURCES** list in lsb.params.

When preempting a job for memory, LSF does not free the memory occupied by the job. Rather, it suspends the job and dispatches another job to the host. It relies on the operating system to swap out the pages of the stopped job as memory of the running job grows.

Default

Not defined (if preemptive scheduling is configured, LSF preempts on job slots only)

PREEMPTION_WAIT_TIME

Syntax

PREEMPTION_WAIT_TIME=*seconds*

Description

You must also specify **PREEMPTABLE_RESOURCES** in lsb.params.

The amount of time LSF waits, after preempting jobs, for preemption resources to become available. Specify at least 300 seconds.

If LSF does not get the resources after this time, LSF might preempt more jobs.

Default

300 (seconds)

PREEXEC_EXCLUDE_HOST_EXIT_VALUES

Syntax

PREEXEC_EXCLUDE_HOST_EXIT_VALUES=all [*~exit_value*] | *exit_value* [*exit_value*] [...]

Description

Specify one or more values (between 1 and 255, but not 99) that corresponds to the exit code your pre-execution scripts exits with in the case of failure. LSF excludes any hosts that attempt to run the pre-exec script and exit with the value specified in **PREEXEC_EXCLUDE_HOST_EXIT_VALUES**.

The exclusion list exists for this job until the **mbatchd** restarts.

Specify more than one value by separating them with a space. 99 is a reserved value. For example, PREEXEC_EXCLUDE_HOST_EXIT_VALUES=1 14 19 20 21.

Exclude values using a "~": PREEXEC_EXCLUDE_HOST_EXIT_VALUES=all ~40

In the case of failures that could be avoided by retrying on the same host, add the retry process to the pre-exec script.

Use in combination with **MAX_PREEXEC_RETRY** in lsb.params to limit the total number of hosts that are tried. In a multicluster environment, use in combination with **LOCAL_MAX_PREEXEC_RETRY** and **REMOTE_MAX_PREEXEC_RETRY**.

Default

None.

PRIVILEGED_USER_FORCE_BKILL

Syntax

PRIVILEGED_USER_FORCE_BKILL=y | Y

Description

If Y, only root or the LSF administrator can successfully run **bkill -r**. For any other users, **-r** is ignored. If not defined, any user can run **bkill -r**.

Default

Not defined.

RC_MAX_REQUESTS

Syntax

RC_MAX_REQUESTS=*integer*

Description

For LSF resource connector. Specifies the maximum number of new instances that can be requested through the resource connector during demand evaluations. Use this parameter to control the speed of cluster growth in the cluster.

Default

300

See also

LSB_RC_UPDATE_INTERVAL in the `lsf.conf` file, which controls the minimum interval between two demand evaluations.

RELAX_JOB_DISPATCH_ORDER

Allows LSF to deviate from standard job prioritization policies to improve cluster utilization.

Syntax

RELAX_JOB_DISPATCH_ORDER=Y | y | N | n

RELAX_JOB_DISPATCH_ORDER=[ALLOC_REUSE_DURATION[[*min*] *max*]]

Description

When this parameter is enabled, LSF allows multiple jobs with common resource requirements to run consecutively on the same allocation. Whenever a job finishes, LSF attempts to quickly replace it with a pending job that has the same resource requirements. To ensure that limits are not violated, LSF selects pending jobs that belong to the same user and have other attributes in common.

Since LSF bypasses most of the standard scheduling logic between jobs, reusing resource allocation can help improve cluster utilization. This improvement is most evident in clusters with several shorter jobs (that is, jobs that run from a few seconds to several minutes).

To ensure that the standard job prioritization policies are approximated, there is a limit on the length of time that each allocation is reusable. LSF automatically sets this time limit to achieve a high level of resource utilization. By default, this reuse time cannot exceed 30 minutes. If you specify a maximum reuse time and an optional minimum reuse time (by using **ALLOC_REUSE_DURATION[]**), LSF adjusts the time limit within this specified range to achieve the highest level of resource utilization.

When jobs from job arrays reuse allocations, or jobs are shorter than the **ALLOC_REUSE_DURATION[]** values, the dispatch order of these jobs may change. This is because jobs are chosen for allocation reuse based on submission time instead of other factors.

Advance reservations are not considered when reusing the job allocation. A job that is placed on a host with advance reservations enabled cannot have its allocation reused. If an advance reservation is created on a host after the job allocation is already made, the allocation can still be reused until the reuse duration is expired or the job is suspended by the advance reservation policy.

This parameter can also be set for an individual queue in `lsb.queues`. If defined, the queue value takes precedence.

Examples

- `RELAX_JOB_DISPATCH_ORDER=Y`

The resource allocation of a finished job can be reused from 0 to 30 minutes.

- `RELAX_JOB_DISPATCH_ORDER=ALLOC_REUSE_DURATION[45]`

The resource allocation of a finished job can be reused from 0 to 45 minutes.

- `RELAX_JOB_DISPATCH_ORDER=ALLOC_REUSE_DURATION[5 45]`

The resource allocation of a finished job can be reused from 5 to 45 minutes.

Default

Set to Y at time of installation. Otherwise, N.

If set to Y, the **ALLOC_REUSE_DURATION[]** value is between 0 and 30 minutes.

Note: A setting of N gives the same behavior seen in LSF, Version 9, and earlier. This behavior changed with LSF, Version 10.

REMOVE_HUNG_JOBS_FOR

Syntax

```
REMOVE_HUNG_JOBS_FOR = runlimit[,wait_time=min] | host_unavail[,wait_time=min]
| runlimit [,wait_time=min]:host_unavail[,wait_time=min] | all[,wait_time=min]
```

Description

Hung jobs are removed under the following conditions:

- **host_unavail:** Hung jobs are automatically removed if the first execution host is unavailable and a timeout is reached as specified by **wait_time** in the parameter configuration. The default value of **wait_time** is 10 minutes.

Hung jobs of any status (RUN, SSUSP, etc.) will be a candidate for removal by LSF when the timeout is reached.

- **runlimit:** Remove the hung job after the job's run limit was reached. You can use the **wait_time** option to specify a timeout for removal after reaching the **runlimit**. The default value of **wait_time** is 10 minutes. For example, if **REMOVE_HUNG_JOBS_FOR** is defined with **runlimit**, **wait_time=5** and **JOB_TERMINATE_INTERVAL** is not set, the job is removed by **mbatchd** 5 minutes after the job **runlimit** is reached.

Hung jobs in RUN status are considered for removal if the **runlimit** + **wait_time** have expired.

For backwards compatibility with earlier versions of LSF, **REMOVE_HUNG_JOBS_FOR = runlimit** is handled as previously: The grace period is 10 mins + MAX(6 seconds, **JOB_TERMINATE_INTERVAL**)

where **JOB_TERMINATE_INTERVAL** is specified in `lsb.params`. The grace period only begins once a job's run limit has been reached.

- **all**: Specifies hung job removal for all conditions (both **runlimit** and **host_unavail**). The hung job is removed when the first condition is satisfied. For example, if a job has a run limit, but it becomes hung because a host is unavailable before the run limit is reached, jobs (running, suspended, etc.) will be removed after 10 minutes after the host is unavailable. Job is placed in EXIT status by **mbatchd**.

For a **host_unavail** condition, **wait_time** count starts from the moment **mbatchd** detects that the host is `unavail`. Running **badmin mbdrestart** or **badmin reconfig** while the timeout is in progress will restart the timeout countdown from 0.

For a **runlimit** condition, **wait_time** is the time that the job in the UNKNOWN state takes to reach the **runlimit**.

Default

Not defined.

REMOTE_MAX_PREEEXEC_RETRY

Syntax

`REMOTE_MAX_PREEEXEC_RETRY=integer`

Description

The maximum number of times to attempt the pre-execution command of a job from the remote cluster.

Valid values

`0 < REMOTE_MAX_PREEEXEC_RETRY < 2147483647`

Default

5

RESIZABLE_JOBS

Allows jobs to be resized or to be autoresizable.

Syntax

`RESIZABLE_JOBS = [Y|N|auto]`

Description

`N|n`: The resizable job feature is disabled. Under this setting, all jobs are not resizable. All **bresize** and **bsub -ar** commands will be rejected with a proper error message.

`Y|y`: Resize is enabled and all jobs are resizable by default. Under this setting, users can run **bresize** commands to cancel pending resource allocation requests for the job or release resources from an existing job allocation, or use **bsub** to submit an autoresizable job.

`auto`: All jobs will be autoresizable.

Resizable jobs must be submitted defining **RESIZABLE_JOBS** as either `auto` or `Y`. If **RESIZABLE_JOBS=auto**, but the administrator changes it to `N` and reconfigures LSF, jobs without job-level `auto` resizable attribute become not autoresizable. For running jobs that are in the middle of notification stage, LSF lets current notification complete and stops scheduling. Changing **RESIZABLE_JOBS** configuration does not affect jobs with job-level autoresizable attribute. (This behavior is same as exclusive job, **bsub -x** and **EXCLUSIVE** parameter in queue level.)

Resizable jobs can have alternative and compound resource requirements. When using **bresize** **release** to release slots from compound resource requirements, you can only release slots represented by the last term of the compound resource requirement. To release slots in earlier terms, run **bresize** **release** repeatedly to release slots in subsequent last terms.

Default

If the parameter is undefined, the default value is N.

See also

RESIZABLE_JOBS in lsb.applications

RESOURCE_RESERVE_PER_TASK

Syntax

Description

RESOURCE_RESERVE_PER_TASK=Y|y|N|n

If set to Y, **mbatchd** reserves resources based on job tasks instead of per-host.

By default, **mbatchd** only reserves resources for parallel jobs on a per-host basis. For example, by default, the command:

```
bsub -n 4 -R "rusage[mem=500]" -q reservation my_job
```

requires the job to reserve 500 MB on each host where the job runs.

Some parallel jobs need to reserve resources based on job tasks, rather than by host. In this example, if per-task reservation is enabled by **RESOURCE_RESERVE_PER_TASK**, the job `my_job` must reserve 500 MB of memory for each job task ($4 \times 500 = 2$ GB) on the host in order to run.

If **RESOURCE_RESERVE_PER_TASK** is set, the following command reserves the resource `my_resource` for all 4 job tasks instead of only 1 on the host where the job runs:

```
bsub -n 4 -R "my_resource > 0 rusage[my_resource=1]" myjob
```

Default

N (Not defined)

When a shared resource is set to the default behavior, the job reserves the resource per *job*. A shared resource can be defined as a License Scheduler resource or a LSF resource.

When a host-based resource is set to the default behavior, the job reserves the resources per *host*.

RUN_JOB_FACTOR

Syntax

RUN_JOB_FACTOR=*number*

Description

Used only with fairshare scheduling. Job slots weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the number of job slots reserved and in use by a user.

This parameter can also be set for an individual queue in `lsb.queues`. If defined, the queue value takes precedence.

lsb.params

Default

3.0

RUN_TIME_DECAY

Syntax

RUN_TIME_DECAY=Y | y | N | n

Description

Used only with fairshare scheduling. Enables decay for run time at the same rate as the decay set by HIST_HOURS for cumulative CPU time and historical run time.

In the calculation of a user's dynamic share priority, this factor determines whether run time is decayed.

This parameter can also be set for an individual queue in `lsb.queue`s. If defined, the queue value takes precedence.

Restrictions

Running `badadmin reconfig` or restarting `mbatchd` during a job's run time results in the decayed run time being recalculated.

When a suspended job using run time decay is resumed, the decay time is based on the elapsed time.

Default

N

RUN_TIME_FACTOR

Syntax

RUN_TIME_FACTOR=*number*

Description

Used only with fairshare scheduling. Run time weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the total run time of a user's running jobs.

This parameter can also be set for an individual queue in `lsb.queue`s. If defined, the queue value takes precedence.

Default

0.7

SBD_SLEEP_TIME

Syntax

SBD_SLEEP_TIME=*seconds*

Description

The interval at which LSF checks the load conditions of each host, to decide whether jobs on the host must be suspended or resumed.

The job-level resource usage information is updated at a maximum frequency of every **SBD_SLEEP_TIME** seconds.

The update is done only if the value for the CPU time, resident memory usage, or virtual memory usage has changed by more than 10 percent from the previous update or if a new process or process group has been created.

The LIM marks the host SBDDOWN if it does not receive the heartbeat in 1 minute. Therefore, setting **SBD_SLEEP_TIME** greater than 60 seconds causes the host to be frequently marked SBDDOWN and triggers mbatchd probe, thus slowing performance.

After modifying this parameter, use **badadmin hrestart -f all** to restart sbatchds and let the modified value take effect.

Default

30 seconds.

SCHED_METRIC_ENABLE

Syntax

SCHED_METRIC_ENABLE=Y | N

Description

Enable scheduler performance metric collection.

Use **badadmin perfmon stop** and **badadmin perfmon start** to dynamically control performance metric collection.

Default

N

SCHED_METRIC_SAMPLE_PERIOD

Syntax

SCHED_METRIC_SAMPLE_PERIOD=*seconds*

Description

Set a default performance metric sampling period in seconds.

Cannot be less than 60 seconds.

Use **badadmin perfmon setperiod** to dynamically change performance metric sampling period.

Default

60 seconds

SCHED_PER_JOB_SORT

Syntax

SCHED_PER_JOB_SORT=Y/N

Description

Enable this parameter to use the per-job sorting feature in scheduler. This feature allows LSF to schedule jobs accurately in one scheduling cycle by adding extra load to scheduler.

For example, jobs 1 to 3 are submitted to LSF as follows:

- **bsub -R "order[slots]" job1**
- **bsub -R "order[slots]" job2**
- **bsub -R "order[slots]" job3**

Each job was requested to be dispatched to the host with the most unused slots. The request is not completely satisfied in one scheduling cycle if per-job sorting is disabled. For example:

- host1 3 slots
- host2 3 slots
- host3 3 slots

Jobs 1 to 3 are dispatched to the same host (for example, host1) in one scheduling cycle because the hosts are only sorted once in one scheduling cycle. If the per-job sorting feature is enabled, the candidate hosts are sorted again before the job is actually scheduled. Therefore, these three jobs are dispatched to each host separately. This result is exactly what you want but the cost is that scheduler performance maybe significantly lower. For example, if there are 5000 hosts in the cluster and the master host is on the machine with 24GB memory and 2 physical CPUs with 2 cores with 6 threads, each job with ! in the ORDER[] section consumes about an extra 10ms of time in one scheduling cycle. If there are many jobs with ! in the ORDER[] section that are controlled by **LSB_MAX_JOB_DISPATCH_PER_SESSION**, then a lot of extra time is consumed in one scheduling cycle.

To get the accurate schedule result without impacting scheduler performance, you can set **JOB_ACCEPT_INTERVAL** as a non-zero value. This is because the hosts do not need to be sorted again by the per-job sorting feature.

Default

N

See also

JOB_ACCEPT_INTERVAL in lsb.params and lsb.queues.

LSB_MAX_JOB_DISPATCH_PER_SESSION in lsf.conf.

SCHEDULER_THREADS**Syntax**

SCHEDULER_THREADS=integer

Description

Set the number of threads the scheduler uses to evaluate resource requirements.

Multithreaded resource evaluation is useful for large scale clusters with large numbers of hosts. The idea is to do resource evaluation for hosts concurrently. For example, there are 6,000 hosts in a cluster, so the scheduler may create six threads to do the evaluation concurrently. Each thread is in charge of 1,000 hosts.

To set an effective value for this parameter, consider the number of available CPUs on the master host, the number of hosts in the cluster, and the scheduling performance metrics. Set the number of threads between 1 and the number of cores in the master host. A value of 0 means that the scheduler does not create any threads to evaluate resource requirements.

Default

0

SECURE_INFODIR_USER_ACCESS

Prevents users from viewing detailed information for jobs that belong to other users.

Syntax

```
SECURE_INFODIR_USER_ACCESS=Y | N | G
```

Description

- By default, (**SECURE_INFODIR_USER_ACCESS=N** or is not defined), any user can view detailed information for jobs that belong to other users in the `lsb.event` and `lsb.acct` files by using the **bhist** or **bacct** commands. Specify Y or G to restrict this access so that regular and administrator users only see their own job information.
- Specify Y to prevent users (including all users except the primary administrator) from accessing other users' job information using the **bhist** or **bacct** commands.

You can control the granularity of the **bjobs** command to specify the information that other users can see by specifying a value for the **SECURE_JOB_INFO_LEVEL** parameter in the `lsb.params` file.

A regular user does not have rights to call the API to get data under `LSB_SHAREDIR/cluster/logdir`, which is readable only by the primary administrator. Regular and administrator users do not have rights to run **bhist -t**. Only the primary administrator has rights to run **bhist -t**.

- Specify G to prevent users (including all users except the primary administrator) from accessing other users' job information.

You can control the granularity of the information that the **bhist** and **bacct** commands display by specifying the **SECURE_JOB_INFO_LEVEL** and **ENABLE_JOB_INFO_BY_ADMIN_ROLE** parameters in the `lsb.params` file.

A regular user does not have rights to call the API to get data under `LSB_SHAREDIR/cluster/logdir`, which is readable only by the primary administrator. Regular and administrator users do not have rights to run **bhist -t**. Only the primary administrator has rights to run **bhist -t**.

The LSF primary administrator can always view job information for all users in the `lsb.event` and `lsb.acct` files, no matter what the setting.

After enabling this feature by setting **SECURE_INFODIR_USER_ACCESS** to Y or G, you must `setuid` of the LSF primary administrator for **bhist** and **bacct** binary under `LSF_BINDIR`. **bhist** and **bacct** will call **mbatchd** to check whether the parameter is set or not when you have `setuid` for **bhist** and **bacct**.

To disable this feature, set **SECURE_INFODIR_USER_ACCESS** to N, then remove the `setuid` for **bhist** and **bacct** binary under `LSF_BINDIR` to prevent **bhist** and **bacct** from calling **mbatchd**. When disabled, the permission to `LSB_SHAREDIR/cluster/logdir` returns to normal after you reconfigure **mbatchd** (by running **badmin reconfig**).

Note:

- This feature is only supported when LSF is installed on a file system that supports the `setuid` bit for files. Therefore, this feature does not work on Windows platforms.
- If **LSB_LOCALDIR** is enabled to duplicate **LSB_SHAREDIR**, **LSB_LOCALDIR** is also readable only by the primary administrator after setting **SECURE_INFODIR_USER_ACCESS=Y**.

Default

N

Related reference

SECURE_JOB_INFO_LEVEL parameter in the `lsb.params` file.

SECURE_JOB_INFO_LEVEL

Syntax

SECURE_JOB_INFO_LEVEL=0 | 1 | 2 | 3 | 4 | 5

Description

Defines an access control level for all users. Specify a level to control which jobs users and administrators (except the primary administrator) can see.

For LSF users, there are three types of jobs:

- The user's own jobs.
- Jobs that belong to other users in the same user group.
- Jobs that do not belong to users in the same user group.

There are two kinds of job information which will be viewed by users:

- Detailed information:

Obtained from **bjobs -l**, **bjobs -UF**, **bjobs -N**, **bjdepinfo**, **bread**, and **bstatus**.

If **SECURE_INFODIR_USER_ACCESS** is set to G, this is also obtained from **bacct -l**, **bacct -UF**, **bhist -b**, **bhist -l**, and **bhist -UF**.

- Summary information:

Obtained from **bjobs** with options other than **-l**, such as **-aps**, **-fwd**, **-p**, **-ss**, **-sum**, **-w**, **-WF**, **-WP**, and **-WL**.

If **SECURE_INFODIR_USER_ACCESS** is set to G, this is also obtained from **bacct** with options other than **-l** and **-UF** (such as **-d**, **-e**, **-q**, and **-w**), and from **bhist** with options other than **-b**, **-l**, and **-UF** (such as **-a**, **-b**, and **-d**).

There are two kinds of user rights which will determine what kind of information a user can view for a job:

- Basic rights: User can see all summary information.
- Detail rights: User can see all detail information.

When a user or admin enters one of the commands to see job information (**bjobs**, **bjdepinfo**, **bread**, or **bstatus**), the **SECURE_JOB_INFO_LEVEL** parameter controls whether they see:

- Summary information for all jobs (level 5).
- No information for any other jobs (level 4).
- Summary information from jobs in the same user group (level 3).
- Detailed and summary information from jobs in the same user group (level 2).
- Detailed and summary information from jobs in the same user group, and summary information from jobs outside their user group (level 1).
- Detailed and summary information for all jobs (level 0).

Regardless of the setting, users can always see summary and detailed information for their own jobs.

If **SECURE_INFODIR_USER_ACCESS** is set to G, the **SECURE_JOB_INFO_LEVEL** parameter also controls whether they see this information for the **bacct** and **bhist** commands.

Note:

- If **SECURE_JOB_INFO_LEVEL** is set to a level greater than 0, LSF checks if **SECURE_INFODIR_USER_ACCESS** is enabled (set to Y or G). If it is not enabled, access to **bjobs** functions will be restricted, but access to **bhist** / **bacct** will be available.

- When using the LSF multicluster capability, the **SECURE_JOB_INFO_LEVEL** definition still applies when a user attempts to view job information from a remote cluster through the **bjobs -m remotecoluster** command. The security level configuration of a specified cluster will take effect.

Interaction with **bsub -G** and **bjobs -o**

- If a user submits a job using **bsub -G**, the job will be treated as a member of the **-G** specified user group (or default user group). For example: UserA belongs to user groups UG1 and UG2. UserA submits Job1 using **bsub -G**:

```
bsub -G UG1
```

UserA submits Job2 without using **bsub -G**. Job1 will be treated as belonging to UG1 only. Job2 will be treated as belonging to UG1 and UG2. The result is that members of UG1 can view both Job1 and Job2 details if they are given access rights to view jobs in the same user group. Members of UG2 can view only Job2 if they are given access rights to view jobs in the same user group.

- If a user has only basic rights, **bjobs -o** returns only values in the basic fields (others display as "-"): Jobid, stat, user, queue, job_name, proj_name, pids, from_host, exec_host, nexec_host, first_host, submit_time, start_time, time_left, finish_time, %complete, cpu_used, slots, mem, swap, forward_cluster, forward_time, run_time.

Limitations

- Administrators may not have permission to see a job, but they can still control a job (for example, kill a job) using the appropriate commands.
- When job information security is enabled, pre-LSF 9.1 **bjobs** and **bjdepinfo** commands will be rejected no matter who issues them because mbatchd cannot get the command user name. A "No job found" message will be returned.
- When job information security is enabled, users may have rights to only view job summary information and no rights to view job detail information. Therefore, a user would see job info when viewing summary info (using **bjobs <jobid>**), but an error (job <jobid> is not found) will be returned when the user tries to view job detail information (using **bjobs -l <jobid>**).

Default

0

SLA_TIMER

Syntax

SLA_TIMER=*seconds*

Description

For EGO-enabled SLA scheduling. Controls how often each service class is evaluated and a network message is sent to EGO communicating host demand.

Valid values

Positive integer between 2 and 21474847

Default

0 (Not defined.)

SSCHED_ACCT_DIR

Syntax

SSCHED_ACCT_DIR=*directory*

Description

Used by IBM Platform Session Scheduler (**ssched**).

A universally accessible and writable directory that will store Session Scheduler task accounting files. Each Session Scheduler session (each **ssched** instance) creates one accounting file. Each file contains one accounting entry for each task. The accounting file is named *job_ID.ssched.acct*. If no directory is specified, accounting records are not written.

Valid values

Specify any string up to 4096 characters long

Default

Not defined. No task accounting file is created.

SSCHED_MAX_RUNLIMIT

Syntax

SSCHED_MAX_RUNLIMIT=*seconds*

Description

Used by IBM Platform Session Scheduler (**ssched**).

Maximum run time for a task. Users can override this value with a lower value. Specify a value greater than or equal to zero (0).

Recommended value

For very short-running tasks, a reasonable value is twice the typical runtime. Because LSF does not release slots allocated to the session until all tasks are completed and **ssched** exits, you should avoid setting a large value for SSCHED_MAX_RUNLIMIT.

Valid values

Specify a positive integer between 0 and 2147483645

Default

600 seconds (10 minutes)

SSCHED_MAX_TASKS

Syntax

SSCHED_MAX_TASKS=*integer*

Description

Used by Session Scheduler (**ssched**).

Maximum number of tasks that can be submitted to Session Scheduler. Session Scheduler exits if this limit is reached. Specify a value greater than or equal to zero (0).

Valid values

Specify a positive integer between 0 and 2147483645

Default

50000 tasks

SSCHED_REQUEUE_LIMIT**Syntax**

SSCHED_REQUEUE_LIMIT=*integer*

Description

Used by Session Scheduler (**ssched**).

Number of times Session Scheduler tries to requeue a task as a result of the REQUEUE_EXIT_VALUES (**ssched -Q**) setting. SSCHED_REQUEUE_LIMIT=0 means never requeue. Specify a value greater than or equal to zero (0).

Valid values

Specify a positive integer between 0 and 2147483645

Default

3 requeue attempts

SSCHED_RETRY_LIMIT**Syntax**

SSCHED_RETRY_LIMIT=*integer*

Description

Used by Session Scheduler (**ssched**).

Number of times Session Scheduler tries to retry a task that fails during dispatch or setup. SSCHED_RETRY_LIMIT=0 means never retry. Specify a value greater than or equal to zero (0).

Valid values

Specify a positive integer between 0 and 2147483645

Default

3 retry attempts

SSCHED_UPDATE_SUMMARY_BY_TASK**Syntax**

SSCHED_UPDATE_SUMMARY_BY_TASK=*integer*

Description

Used by Platform Session Scheduler (**ssched**).

Update the Session Scheduler task summary via **bpost** after the specified number of tasks finish. Specify a value greater than or equal to zero (0).

If both `SSCHED_UPDATE_SUMMARY_INTERVAL` and `SSCHED_UPDATE_SUMMARY_BY_TASK` are set to zero (0), **bpost** is not run.

Valid values

Specify a positive integer between 0 and 2147483645

Default

0

See also

`SSCHED_UPDATE_SUMMARY_INTERVAL`

SSCHED_UPDATE_SUMMARY_INTERVAL

Syntax

`SSCHED_UPDATE_SUMMARY_INTERVAL=seconds`

Description

Used by Platform Session Scheduler (**ssched**).

Update the Session Scheduler task summary via **bpost** after the specified number of seconds. Specify a value greater than or equal to zero (0).

If both `SSCHED_UPDATE_SUMMARY_INTERVAL` and `SSCHED_UPDATE_SUMMARY_BY_TASK` are set to zero (0), **bpost** is not run.

Valid values

Specify a positive integer between 0 and 2147483645

Default

60 seconds

See also

`SSCHED_UPDATE_SUMMARY_BY_TASK`

STRICT_UG_CONTROL

Syntax

`STRICT_UG_CONTROL=Y | N`

Description

When `STRICT_UG_CONTROL=Y` is defined:

- Jobs submitted with `-G usergroup` specified can only be controlled by the usergroup administrator of the specified user group.

- user group administrators can be defined for user groups with all as a member

After adding or changing **STRICT_UG_CONTROL** in `lsb.params`, use `badadmin reconfig` to reconfigure your cluster.

Default

N (Not defined.)

See also

DEFAULT_USER_GROUP, **ENFORCE_ONE_UG_LIMIT**, **ENFORCE_UG_TREE**

STRIPING_WITH_MINIMUM_NETWORK

Syntax

STRIPING_WITH_MINIMUM_NETWORK=*y* | *n*

Description

For LSF IBM Parallel Environment (PE) integration. Specifies whether or not nodes which have more than half of their networks in READY state are considered for PE jobs with `type=sn_all` specified in the network resource requirements (in the `bsub -network` option or the `NETWORK_REQ` parameter in `lsb.queues` or `lsb.applications`). This makes certain that at least one network is UP and in READY state between any two nodes assigned for the job.

When set to `y`, the nodes which have more than half the minimum number of networks in the READY state are considered for **sn_all** jobs. If set to `n`, only nodes which have all networks in the READY state are considered for **sn_all** jobs.

Note: `LSF_PE_NETWORK_NUM` must be defined with a value greater than 0 for `STRIPING_WITH_MINIMUM_NETWORK` to take effect.

Example

In a cluster with 8 networks, due to hardware failure, only 3 networks are ok on `hostA`, and 5 networks are ok on `hostB`. If `STRIPING_WITH_MINIMUM_NETWORK=n`, an **sn_all** job cannot run on either `hostA` or `hostB`. If `STRIPING_WITH_MINIMUM_NETWORK=y`, an **sn_all** job can run on `hostB`, but it cannot run on `hostA`.

Default

`n`

SUB_TRY_INTERVAL

Syntax

SUB_TRY_INTERVAL=*integer*

Description

The number of seconds for the requesting client to wait before resubmitting a job. This is sent by `mbatchd` to the client.

Default

60 seconds

See also

MAX_PEND_JOBS

SYSTEM_MAPPING_ACCOUNT

Syntax

SYSTEM_MAPPING_ACCOUNT=*user_account*

Description

Enables Windows workgroup account mapping, which allows LSF administrators to map all Windows workgroup users to a single Windows system account, eliminating the need to create multiple users and passwords in LSF. Users can submit and run jobs using their local user names and passwords, and LSF runs the jobs using the mapped system account name and password. With Windows workgroup account mapping, all users have the same permissions because all users map to the same system account.

To specify the user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).

Define this parameter for LSF Windows Workgroup installations only.

Default

Not defined

TRACK_ELIGIBLE_PENDINFO

Allows LSF to determine whether a pending job is eligible for scheduling.

Syntax

TRACK_ELIGIBLE_PENDINFO=Y | y | N | n

Description

Set to Y or y to allow LSF to determine whether a pending job is eligible for scheduling, and to use eligible pending time instead of total pending time to determine job priority for the following time-based scheduling policies:

- Automatic job priority escalation: Only increases job priority of jobs that have been in an eligible pending state instead of pending state for the specified period of time.
- Absolute priority scheduling (APS): The JPRIORITY subfactor for the APS priority calculation uses the amount of time that the job spent in an eligible pending state instead of the total pending time.

A job that is in an eligible pending state is a job that LSF would normally select for resource allocation, but is currently pending because its priority is lower than other jobs. It is a job that is eligible for scheduling and will be run if there are sufficient resources to run it.

An ineligible pending job remains pending even if there are enough resources to run it and is therefore ineligible for scheduling. Reasons for a job to remain pending, and therefore be in an ineligible pending state, include the following:

- The job has a start time constraint (specified with the -b option)
- The job is suspended while pending (in a PSUSP state).
- The queue of the job is made inactive by the administrator or by its time window.
- The job's dependency conditions are not satisfied.
- The job cannot fit into the run time window (**RUN_WINDOW**)
- Delayed scheduling is enabled for the job (**NEW_JOB_SCHED_DELAY** is greater than zero)
- The job's queue or application profile does not exist.

A job that is not under any of the ineligible pending state conditions is treated as an eligible pending job. In addition, for chunk jobs in WAIT status, the time spent in the WAIT status is counted as eligible pending time.

When **mbatchd** is restarted, some ineligible pending time may be lost since it is recovered from the snapshot file, which is dumped periodically at set intervals. The lost time period is counted as eligible pending time under such conditions.

In multicluster job forwarding mode, if the **MC_SORT_BY_SUBMIT_TIME** parameter is enabled, LSF counts all pending time before the job is forwarded as eligible for a forwarded job in the execution cluster.

Default

Set to Y at time of installation for the PARALLEL configuration template. Otherwise, N (disabled).

See also

ELIGIBLE_PENDINFO_SNAPSHOT_INTERVAL

USE_SUSP_SLOTS

Syntax

USE_SUSP_SLOTS=Y | N

Description

If **USE_SUSP_SLOTS=Y**, allows jobs from a low priority queue to use slots held by suspended jobs in a high priority queue, which has a preemption relation with the low priority queue.

Set **USE_SUSP_SLOTS=N** to prevent low priority jobs from using slots held by suspended jobs in a high priority queue, which has a preemption relation with the low priority queue.

Default

Y

lsb.queues

The `lsb.queues` file defines batch queues. Numerous controls are available at the queue level to allow cluster administrators to customize site policies.

This file is optional. If no queues are configured, LSF creates a queue that is named `default`, with all parameters set to default values.

This file is installed by default in `LSB_CONFDIR/cluster_name/configdir`.

Changing lsb.queues configuration

After you change `lsb.queues`, run **badmin reconfig** to reconfigure `mbatchd`.

Some parameters, such as run window and runtime limit, do not take effect immediately for running jobs unless you run **mbatchd restart** or **sbatchd restart** on the job execution host.

lsb.queues structure

Each queue definition begins with the line `Begin Queue` and ends with the line `End Queue`. The queue name must be specified; all other parameters are optional.

#INCLUDE

Syntax

```
#INCLUDE "path-to-file"
```

Description

Inserts a configuration setting from another file to the current location. Use this directive to dedicate control of a portion of the configuration to other users or user groups by providing write access for the included file to specific users or user groups, and to ensure consistency of configuration file settings in different clusters (if you are using the LSF multicluster capability).

See more information on shared configuration file content in *Administering IBM Spectrum LSF*

#INCLUDE can be inserted anywhere in the local configuration file.

Default

Not defined.

ADMINISTRATORS

Specifies a space-separated list of queue administrators. Queue administrators can operate on any user job in the queue, and on the queue itself.

Syntax

```
ADMINISTRATORS=user_name | user_group ...
```

Description

To specify a Windows user account or user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_name* or *DOMAIN_NAME\user_group*).

Default

Not defined. You must be a cluster administrator to operate on this queue.

APS_PRIORITY

Specifies calculation factors for absolute priority scheduling (APS). Pending jobs in the queue are ordered according to the calculated APS value.

Syntax

```
APS_PRIORITY=WEIGHT[[factor, value][subfactor, value]...]...] LIMIT[[factor, value][subfactor, value]...]...] GRACE_PERIOD[[factor, value][subfactor, value]...]...]
```

Description

If weight of a subfactor is defined, but the weight of parent factor is not defined, the parent factor weight is set as 1.

The WEIGHT and LIMIT factors are floating-point values. Specify a *value* for GRACE_PERIOD in seconds (*values*), minutes (*valuem*), or hours (*valueh*).

The default unit for grace period is hours.

The following are the names of the factors and subfactors to specify:

Factors	Subfactors	Metric
FS (user-based fairshare factor)	The existing fairshare feature tunes the dynamic user priority	<p>The fairshare factor automatically adjusts the APS value based on dynamic user priority.</p> <p>The FAIRSHARE parameter must be defined in the queue. The FS factor is ignored for non-fairshare queues.</p> <p>The FS factor is influenced by the following fairshare parameters that are defined in the <code>lsb.queues</code> or <code>lsb.params</code> file:</p> <ul style="list-style-type: none"> • CPU_TIME_FACTOR • FWD_JOB_FACTOR • RUN_TIME_FACTOR • RUN_JOB_FACTOR • HIST_HOURS
RSRC (resource factors)	PROC	Requested tasks are the maximum of bsub -n min_task, max_task , the min of bsub -n min , or the value of the TASKLIMIT parameter in the <code>lsb.queues</code> file.
	MEM	<p>Total real memory requested (in MB or in units set in the LSF_UNIT_FOR_LIMITS parameter in the <code>lsf.conf</code> file).</p> <p>Memory requests appearing to the right of a <code> </code> symbol in a usage string are ignored in the APS calculation.</p> <p>For multi-phase memory reservation, the APS value is based on the first phase of reserved memory.</p>
	SWAP	<p>Total swap space requested (in MB or in units set in the LSF_UNIT_FOR_LIMITS parameter in the <code>lsf.conf</code> file).</p> <p>As with MEM, swap space requests appearing to the right of a <code> </code> symbol in a usage string are ignored.</p>

Factors	Subfactors	Metric
WORK (job attributes)	JPRIORITY	<p>The job priority that is specified by:</p> <ul style="list-style-type: none"> • Default that is specified by half of the value of the MAX_USER_PRIORITY parameter in the <code>lsb.params</code> file • Users with bsub -sp or bmod -sp • Automatic priority escalation with the JOB_PRIORITY_OVER_TIME parameter in the <code>lsb.params</code> file <p>If the TRACK_ELIGIBLE_PENDINFO parameter in the <code>lsb.params</code> file is set to Y or y, LSF increases the job priority for pending jobs as long as it is eligible for scheduling. LSF does not increase the job priority for ineligible pending jobs.</p>
	QPRIORITY	The priority of the submission queue.
APP		Set the priority factor at the application profile level by specifying the PRIORITY parameter in the <code>lsb.applications</code> file. The APP_PRIORITY factor is added to the calculated APS value to change the factor value. The APP_PRIORITY factor applies to the entire job.
USER		Set the priority factor for users by specifying the PRIORITY parameter in the User section of the <code>lsb.users</code> file. The USER_PRIORITY factor is added to the calculated APS value to change the factor value. The USER_PRIORITY factor applies to the entire job.

Factors	Subfactors	Metric
UG		Set the priority factor for user groups by specifying the PRIORITY parameter in the UserGroup section of the <code>lsb.users</code> file. The <code>UG_PRIORITY</code> factor is added to the calculated APS value to change the factor value. The <code>UG_PRIORITY</code> factor applies to the entire job. LSF uses the priority of the user group as specified in the bsub -G option.
ADMIN		Administrators use bmod -aps to set this subfactor value for each job. A positive value increases the APS. A negative value decreases the APS. The <code>ADMIN</code> factor is added to the calculated APS value to change the factor value. The <code>ADMIN</code> factor applies to the entire job. You cannot configure separate weight, limit, or grace period factors. The <code>ADMIN</code> factor takes effect as soon as it is set.

For example, the following sets a grace period of 10 hours for the `MEM` factor, 10 minutes for the `JPRIORITY` factor, 10 seconds for the `QPRIORITY` factor, and 10 hours (default) for the `RSRC` factor:

```
GRACE_PERIOD[[MEM,10h] [JPRIORITY, 10m] [QPRIORITY,10s] [RSRC, 10]]
```

You cannot specify 0 (zero) for the `WEIGHT`, `LIMIT`, and `GRACE_PERIOD` of any factor or subfactor.

APS queues cannot configure cross-queue fairshare (**FAIRSHARE_QUEUES**). The **QUEUE_GROUP** parameter replaces `FAIRSHARE_QUEUES`, which is obsolete in LSF 7.0.

Suspended (**bstop**) jobs and migrated jobs (**bmig**) are always scheduled before pending jobs. For migrated jobs, LSF keeps the existing job priority information.

If `LSB_REQUEUE_TO_BOTTOM` and `LSB_MIG2PEND` are configured in `lsf.conf`, the migrated jobs keep their APS information. When `LSB_REQUEUE_TO_BOTTOM` and `LSB_MIG2PEND` are configured, the migrated jobs need to compete with other pending jobs based on the APS value. To reset the APS value, use **brequeue**, not **bmig**.

Default

Not defined

BACKFILL

Enables backfill scheduling for the queue.

Syntax

BACKFILL=Y | N

Description

Set this parameter as Y to enable backfill scheduling for the queue.

A possible conflict exists if **BACKFILL** and **PREEMPTION** are specified together. If **PREEMPT_JOBTYPE = BACKFILL** is set in the `lsb.params` file, a backfill queue can be preemptable. Otherwise, a backfill queue cannot be preemptable. If **BACKFILL** is enabled, do not also specify **PREEMPTION = PREEMPTABLE**.

BACKFILL is required for interruptible backfill queues (**INTERRUPTIBLE_BACKFILL=seconds**).

When **MAX_SLOTS_IN_POOL**, **SLOT_RESERVE**, and **BACKFILL** are defined for the same queue, jobs in the queue cannot backfill with slots that are reserved by other jobs in the same queue.

Default

Not defined. No backfilling.

CHKPNT

Enables automatic checkpointing for the queue. All jobs that are submitted to the queue are checkpointable.

Syntax

```
CHKPNT=chkpnt_dir [chkpnt_period]
```

Description

The checkpoint directory is the directory where the checkpoint files are created. Specify an absolute path or a path relative to CWD, do not use environment variables.

Specify the optional checkpoint period in minutes.

You can checkpoint only running members of a chunk job.

If checkpoint-related configuration is specified in both the queue and an application profile, the application profile setting overrides queue level configuration.

Checkpoint-related configuration that is specified in the queue, application profile, and at job level has the following effect:

- Application-level and job-level parameters are merged. If the same parameter is defined at both job-level and in the application profile, the job-level value overrides the application profile value.
- The merged result of job-level and application profile settings override queue-level configuration.

To enable checkpointing of MultiCluster jobs, define a checkpoint directory in both the send-jobs and receive-jobs queues (**CHKPNT** in `lsb.queues`), or in an application profile (**CHKPNT_DIR**, **CHKPNT_PERIOD**, **CHKPNT_INITPERIOD**, **CHKPNT_METHOD** in `lsb.applications`) of both submission cluster and execution cluster. LSF uses the directory that is specified in the execution cluster.

To make a MultiCluster job checkpointable, both submission and execution queues must enable checkpointing, and the application profile or queue setting on the execution cluster determines the checkpoint directory. Checkpointing is not supported if a job runs on a leased host.

The file path of the checkpoint directory can contain up to 4000 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

Default

Not defined

CHUNK_JOB_SIZE

Enables job chunking and specifies the maximum number of jobs that are allowed to be dispatched together in a chunk.

Syntax

CHUNK_JOB_SIZE=*integer*

Description

Specify a positive integer greater than 1.

The ideal candidates for job chunking are jobs that have the same host and resource requirements and typically take 1 - 2 minutes to run.

Job chunking can have the following advantages:

- Reduces communication between sbatchd and mbatchd and reduces scheduling performance in mbschd.
- Increases job throughput in mbatchd and CPU usage on the execution hosts.

However, throughput can deteriorate if the chunk job size is too large. Performance might decrease on queues with CHUNK_JOB_SIZE greater than 30. Evaluate the chunk job size on your own systems for best performance.

With MultiCluster job forwarding model, this parameter does not affect MultiCluster jobs that are forwarded to a remote cluster.

Compatibility

This parameter is ignored in interactive queues (**INTERACTIVE=ONLY** parameter).

- Interactive (**INTERACTIVE=ONLY** parameter)

If **CHUNK_JOB_DURATION** is set in `lsb.params`, chunk jobs are accepted as long as the CPULIMIT, RUNLIMIT, or **RUNTIME** values do not exceed the **CHUNK_JOB_DURATION**.

Example

The following example configures a queue that is named chunk, which dispatches up to four jobs in a chunk:

```
Begin Queue
QUEUE_NAME      = chunk
PRIORITY        = 50
CHUNK_JOB_SIZE  = 4
End Queue
```

Default

Not defined

COMMITTED_RUN_TIME_FACTOR

Specifies the committed runtime weighting factor. Used only with fairshare scheduling.

Syntax

COMMITTED_RUN_TIME_FACTOR=*number*

Description

In the calculation of a user dynamic priority, this factor determines the relative importance of the committed run time in the calculation. If the `-W` option of **bsub** is not specified at job submission and a **RUNLIMIT** is not set for the queue, the committed run time is not considered.

If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Valid values

Any positive number between 0.0 and 1.0

Default

Not defined.

CONTAINER**Syntax**

```
CONTAINER=docker[image(image_name) options(docker_run_options)]
```

```
CONTAINER=nvidia-docker[image(image_name) options(docker_run_options)]
```

```
CONTAINER=shifter[image(image_name) options(container_options)]
```

```
CONTAINER=singularity[image(image_name) options(container_options)]
```

Description

Enables LSF to use a Docker, NVIDIA Docker, Shifter, or Singularity container for jobs that are submitted to this queue. This parameter uses the following keywords:

docker | nvidia-docker | shifter | singularity

Required. Use one of these keywords to specify the type of container to use for jobs that are submitted to this queue.

image

Required. This keyword specifies the image name that is used in running jobs.

For Docker and NVIDIA Docker jobs, use the **\$LSB_CONTAINER_IMAGE** environment variable to allow users to specify the image name for Docker and NVIDIA Docker container jobs at job submission time. At job submission time, users can specify a specific image name that is in the specified repository server by specifying a value for the **\$LSB_CONTAINER_IMAGE** environment variable.

options

Optional. This keyword specifies the job run options for the container.

To enable a pre-execution script to run, specify an at sign (@) and a full file path to the script, which the execution host must be able to access. Before the container job runs, LSF runs this script with LSF administrator privileges. While the script is running, the jobs' environment variables are passed to the script. When the script finishes running, the output is used as container startup options. The script must provide this output on one line. The method of processing the container job depends on the result of the pre-execution script:

- If the pre-execution script failed, the container job exits with the same exit code from the script. In addition, an external status message is sent to inform the user that the job exited because of script execution failure.
- If the execution of the script is successful but the output contains more than 512 options, LSF only keeps the first 512 options, and the remaining options are ignored.
- If the execution of the script is successful and the output is valid, the output is part of the container job running options. The position of the output from the script in the options is exactly where the user configured the script in the options field.

For Docker and NVIDIA Docker containers, this keyword specifies the Docker job run options for the **docker run** command, which are passed to the job container.

Note:

- Before you specify the Docker job run options, make sure that these options work with the **docker run** command in the command line.
- The `--cgroup-parent`, `--user` (or `-u`), and `--name` options are reserved for LSF internal use. Do not use these options in the options keyword configuration, otherwise the job fails.
If you specified a pre-execution script and the output of this script contains `--cgroup-parent`, `--user`, `-u`, or `--name`, the container job also fails.
- The `-w` and `--ulimit` options are automatically set for LSF. Do not use these options in the options keyword configuration because these specifications override the LSF settings.
- The `-v` option is automatically used by LSF to mount the working directories that LSF requires, such as the current working directory, job spool directory, destination file for the **bsub -f** command, tmp directory, the LSF_TOP, and the checkpoint directory on demand.
- You can configure the `--rm` option in the options keyword configuration to automatically remove containers after the job is done.
- Limitation: If you use the `-d` option, LSF incorrectly gets the status of the Docker jobs as DONE.

For Shifter containers, this keyword specifies the Shifter job run options for the **shifter** command, which are passed to the job container.

Note:

- Run **shifter --help** in the command line to view the options that the **shifter** command supports.
- Before you specify the Shifter job run options, make sure that these options work with the **shifter** command in the command line.
- The `$LD_LIBRARY_PATH` directory is cleaned according to the setuid bit that Shifter uses to work. Therefore, for programs that depend on `$LD_LIBRARY_PATH` to work (such as **openmpi**), ensure that the setuid bit can be properly set inside the container by adding `LD_LIBRARY_PATH` to the **siteEnvAppend** section of the `udiRoot.conf` file.

For Singularity containers, this keyword specifies the Singularity job run options for the **singularity exec** command, which are passed to the job container.

Note:

- Run **singularity exec --help** in the command line to view the options that the **singularity** command supports.
- Before you specify the Singularity job run options, make sure that these options work with the **singularity exec** command in the command line.
- The `$LD_LIBRARY_PATH` directory is cleaned according to the setuid bit that Singularity uses to work. Therefore, for programs that depend on `$LD_LIBRARY_PATH` to work (such as **openmpi**), ensure that the setuid bit can be properly set inside the container by adding `LD_LIBRARY_PATH` to the `ld.so.conf` file and run the **ldconfig** command.

Examples

To specify an Ubuntu image for use with container jobs without specifying any optional keywords,

```
Begin Queue
NAME = dockerrq
CONTAINER = docker[image(repository.example.com:5000/file/path/ubuntu:latest)]
DESCRIPTION = Docker User Service
End Queue
```

```
Begin Queue
NAME = ndockerrq
```

```
CONTAINER = nvidia-docker[image(repository.example.com:5000/file/path/ubuntu:latest)]
DESCRIPTION = NVIDIA Docker User Service
End Queue
```

```
Begin Queue
NAME = shifterq
CONTAINER = shifter[image(ubuntu:latest)]
DESCRIPTION = Shifter User Service
End Queue
```

```
Begin Queue
NAME = singq
CONTAINER = singularity[image(/file/path/ubuntu.img)]
DESCRIPTION = Singularity User Service
End Queue
```

To specify a pre-execution script in the `/share/usr/` directory, which generates the container startup options,

```
Begin Queue
NAME = dockeroptions
CONTAINER = docker[image(repository.example.com:5000/file/path/ubuntu:latest) options(@/share/usr/docker-options.sh)]
DESCRIPTION = Docker User Service with pre-execution script for options
End Queue
```

```
Begin Queue
NAME = ndockeroptions
CONTAINER = nvidia-docker[image(repository.example.com:5000/file/path/ubuntu:latest) options(@/share/usr/ndocker-options.sh)]
DESCRIPTION = NVIDIA Docker User Service with pre-execution script for options
End Queue
```

```
Begin Queue
NAME = shifteroptions
CONTAINER = shifter[image(ubuntu:latest) options(@/share/usr/shifter-options.sh)]
DESCRIPTION = Shifter User Service
End Queue
```

```
Begin Queue
NAME = singqoptions
CONTAINER = singularity[image(/file/path/ubuntu.img) options(@/share/usr/sing-options.sh)]
DESCRIPTION = Singularity User Service
End Queue
```

- For sequential jobs, specify the following **CONTAINER** parameter value for LSF to automatically remove containers after the job is done:

```
CONTAINER=docker[image(image-name) options(--rm)]
```

- For parallel jobs, the network and IPC must work across containers to make **blaunch** work. The execution user ID and user name mapping file must be mounted to the container for **blaunch** authentication.

Therefore, specify the following **CONTAINER** parameter value for LSF to configure the container IPC and network parameters so that **blaunch** can work across multiple containers, to configure the container password file for **blaunch** authentication, and automatically remove containers after the job is done:

```
CONTAINER=docker[image(image-name) options(--rm --network=host --ipc=host -v /path/to/my/passwd:/etc/passwd)]
```

The passwd file must be in the standard format for UNIX and Linux password files, such as the following format:

```
user1:x:10001:10001:::
user2:x:10002:10002:::
```

- To allow users to specify image names for Docker and NVIDIA Docker container jobs at job submission time, use the **\$LSB_CONTAINER_IMAGE** environment variable as the image name when specifying the **image** keyword.

For example, when defining the **CONTAINER** parameter for the `udockerGPU` queue, add the **\$LSB_CONTAINER_IMAGE** environment variable to the image specification:

```
Begin Queue
NAME = udockerGPU
CONTAINER = docker[image(repository.example.com:5000/$LSB_CONTAINER_IMAGE) \
              options(--rm --net=host --ipc=host -v /gpus/u/fred:/data )]
DESCRIPTION = Docker User Service
End Queue
```

Specify a container image name (such as `ubuntu`) at the job submission time by setting the **\$LSB_CONTAINER_IMAGE** environment using one of the following methods:

- Specify the **\$LSB_CONTAINER_IMAGE** environment variable according to your shell environment:
 - In **cs**h or **tc**sh:


```
setenv LSB_CONTAINER_IMAGE ubuntu
```
 - In **sh**, **ksh**, or **bash**:


```
export LSB_CONTAINER_IMAGE=ubuntu
```
- Use the **bsub -env** option:


```
bsub -env LSB_CONTAINER_IMAGE=ubuntu -q udocker a.out -in in.dat -out out.dat
```
- Use an **esub** script to set the `LSB_CONTAINER_IMAGE` environment variable, then call the **esub** with the **bsub** command.

For example, create an **esub.docker** script in the `$LSF_SERVERDIR` directory with the following contents:

```
#!/bin/sh
exec 1>&2
echo "LSB_CONTAINER_IMAGE=\"$1\" " >> $LSB_SUB_MODIFY_ENVFILE
```

Submit a job to call the **esub.docker** script by running the following command:

```
bsub -a "docker(ubuntu)" -q dockeri a.out -in in.dat -out out.dat
```

Default

Undefined

CORELIMIT

Specifies the per-process core file size limit for all job processes from this queue.

Syntax

`CORELIMIT=integer`

Description

Specify this parameter to place a per-process hard core file size limit, in KB, for all of the processes that belong to a job from this queue (see **getrlimit(2)**).

Default

Unlimited

CPU_FREQUENCY

Specifies the CPU frequency for a queue.

Syntax

CPU_FREQUENCY=[*float_number*][*unit*]

Description

All jobs submit to the queue require the specified CPU frequency. Value is a positive float number with units (GHz, MHz, or KHz). If no units are set, the default is GHz.

You can also use **bsub -freq** to set this value.

The submission value overwrites the application profile value, and the application profile value overwrites the queue value.

Default

Not defined (Nominal CPU frequency is used)

CPULIMIT

Specifies the maximum normalized CPU time and, optionally, the default normalized CPU time that is allowed for all processes of jobs that run in this queue. The name of a host or host model specifies the CPU time normalization host to use.

Syntax

CPULIMIT=[*default_limit*] *maximum_limit*

where *default_limit* and *maximum_limit* are defined by the following formula:

[*hour:*]*minute*[/*host_name* | /*host_model*]

Description

Limits the total CPU time the job can use. This parameter is useful for preventing runaway jobs or jobs that use up too many resources.

When the total CPU time for the whole job reaches the limit, a SIGXCPU signal is sent to all processes that belong to the job. If the job has no signal handler for SIGXCPU, the job is killed immediately. If the SIGXCPU signal is handled, blocked, or ignored by the application, then after the grace period expires, LSF sends SIGINT, SIGTERM, and SIGKILL to the job to kill it.

If a job dynamically creates processes, the CPU time that is used by these processes is accumulated over the life of the job.

Processes that exist for fewer than 30 seconds might be ignored.

By default, if a default CPU limit is specified, jobs submitted to the queue without a job-level CPU limit are killed when the default CPU limit is reached.

If you specify only one limit, it is the maximum, or hard, CPU limit. If you specify two limits, the first one is the default, or soft, CPU limit, and the second one is the maximum CPU limit.

If no host or host model is given with the CPU time, LSF uses the default CPU time normalization host that is defined at the queue level (DEFAULT_HOST_SPEC in `lsb.queues`) if it is configured. Otherwise, the default CPU time normalization host that is defined at the cluster level (DEFAULT_HOST_SPEC in `lsb.params`) is used if it is configured. Otherwise, the host with the largest CPU factor (the fastest host in the cluster) is used.

Because sbatchd periodically checks whether a CPU time limit was exceeded, a Windows job that runs under a CPU time limit can exceed that limit by up to SBD_SLEEP_TIME.

On UNIX systems, the CPU limit can be enforced by the operating system at the process level.

You can define whether the CPU limit is a per-process limit that is enforced by the OS or a per-job limit enforced by LSF with `LSB_JOB_CPULIMIT` in `lsf.conf`.

Jobs that are submitted to a chunk job queue are not chunked if `CPULIMIT` is greater than 30 minutes.

Default

Unlimited

CPU_TIME_FACTOR

Specifies the CPU time weighting factor. Used only with fairshare scheduling.

Syntax

`CPU_TIME_FACTOR=number`

Description

In the calculation of a user dynamic share priority, this factor determines the relative importance of the cumulative CPU time that is used by a user's jobs.

If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Default

0.7

CSM_REQ

Specifies the required values for the CSM **bsub** job submission options. These settings override job level CSM options and append system level allocation flags to the job level allocation flags.

Syntax

`CSM_REQ=[j|sm=y|n|d][:step_cgroup=y|n][[:-core_isolation 0|1|2|3|4|5|6]
[:cn_mem=mem_value]][:alloc_flags "flag1 [flag2 ...]][:smt=smt_value]"`

Description

Use a colon (:) to separate multiple CSM job options. The options can appear in any order and none are required. For the `alloc_flags` keyword, specify an alphanumeric string of flags and separate multiple flags with a space. The string cannot contain a colon (:).

Example

```
CSM_REQ=jsm=n:step_cgroup=y:-core_isolation 3:cn_mem=1024
```

Default

Not defined.

DATALIMIT

Specifies the per-process data segment size limit for all job processes from this queue.

Syntax

`DATALIMIT=[default_limit] maximum_limit`

Description

Set this parameter to place a per-process data segment size limit, in KB, for all of the processes that belong to a job from this queue (see **getrlimit(2)**).

By default, if a default data limit is specified, jobs submitted to the queue without a job-level data limit are killed when the default data limit is reached.

If you specify only one limit, it is the maximum, or hard, data limit. If you specify two limits, the first one is the default, or soft, data limit, and the second one is the maximum data limit.

Default

Unlimited

DATA_TRANSFER

Configures the queue as a data transfer queue for LSF data manager.

Syntax

DATA_TRANSFER=Y | N

Description

The **DATA_TRANSFER=Y** parameter enables a queue for data transfer through LSF data manager.

Only one queue in a cluster can be a data transfer queue. Any transfer jobs that are submitted by the data manager go to this queue. If the `lsf.datamanager` file exists, then at least one queue must define the **DATA_TRANSFER** parameter. If this parameter is set, a corresponding `lsf.datamanager` file must exist.

Regular jobs that are submitted to this queue through **bsub** are rejected.

Use the **bstop**, **brresume**, and **bkill** commands to stop, resume, and kill your own transfer jobs in a data transfer queue. LSF administrators and queue administrators can additionally use the **btop** and **bbot** commands to move transfer jobs in the queue. All other commands on jobs in a data transfer queue are rejected. You cannot use the **bswitch** command to switch jobs from other queues to a data transfer queue.

If you change this parameter, LSF data manager transfer jobs that were in the previous queue remain in that queue and are scheduled and run as normal. The LSF data manager is notified of their success or failure.

The following queue parameters cannot be used together with a queue that defines the **DATA_TRANSFER** parameter:

- INTERACTIVE=ONLY
- RCVJOBS_FROM
- MAX_RSCHED_TIME
- SUCCESS_EXIT_VALUES
- RERUNNABLE

A data transfer queue cannot appear in the list of default queues that are defined by the **DEFAULT_QUEUE** parameter in the `lsb.params` file. Jobs that are submitted to the data transfer queue are not attached to the application specified by the **DEFAULT_APPLICATION** parameter in the `lsb.params` file.

Default

N

DEFAULT_EXTSCHEDED

Specifies default external scheduling options for the queue.

Syntax

DEFAULT_EXTSCHEDED=*external_scheduler_options*

Description

-extsched options on the **bsub** command are merged with DEFAULT_EXTSCHEDED options, and -extsched options override any conflicting queue-level options set by DEFAULT_EXTSCHEDED.

Default

Not defined

DEFAULT_HOST_SPEC

Specifies the default CPU time normalization host for the queue.

Syntax

DEFAULT_HOST_SPEC=*host_name* / *host_model*

Description

The CPU factor of the specified host or host model is used to normalize the CPU time limit of all jobs in the queue, unless the CPU time normalization host is specified at the job level.

Default

Not defined. The queue uses the DEFAULT_HOST_SPEC defined in `lsb.params`. If DEFAULT_HOST_SPEC is not defined in either file, LSF uses the fastest host in the cluster.

DESCRIPTION

Specifies a description of the job queue that is displayed by **bqueues -l**.

Syntax

DESCRIPTION=*text*

Description

Use a description that clearly describes the service features of this queue to help users select the proper queue for each job.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (`\`). The maximum length for the text is 512 characters.

DISPATCH_BY_QUEUE

This parameter is obsolete in LSF Version 10.1 Fix Pack 10 and is replaced by the **JOB_DISPATCH_PACK_SIZE** parameter in the `lsb.params` file.

Syntax

DISPATCH_BY_QUEUE=Y|y|N|n

Description

Enables the scheduling decision for the specified queue to be published without waiting for the whole scheduling session to finish.

Set this parameter to increase queue responsiveness. The scheduling decision for the jobs in the specified queue is final and these jobs cannot be preempted within the same scheduling cycle.

Tip: Set this parameter only for your highest priority queue (such as for an interactive queue) to ensure that this queue has the highest responsiveness.

Default

N

DISPATCH_ORDER

Defines an ordered cross-queue fairshare set, which indicates that jobs are dispatched according to the order of queue priorities first, then user fairshare priority.

Syntax

DISPATCH_ORDER=QUEUE

Description

By default, a user has the same priority across the master and subordinate queues. If the same user submits several jobs to these queues, user priority is calculated by taking into account all the jobs the user submits across the master-subordinate set.

If DISPATCH_ORDER=QUEUE is set in the master queue, jobs are dispatched according to queue priorities first, then user priority. Jobs from users with lower fairshare priorities who have pending jobs in higher priority queues are dispatched before jobs in lower priority queues. This behavior avoids having users with higher fairshare priority from getting jobs that are dispatched from low-priority queues.

Jobs in queues with the same priority are dispatched according to user priority.

Queues that are not part of the cross-queue fairshare can have any priority; they are not limited to fall outside of the priority range of cross-queue fairshare queues.

Default

Not defined

DISPATCH_WINDOW

Specifies the time windows in which jobs from this queue are dispatched. After jobs are dispatched, they are no longer affected by the dispatch window.

Syntax

DISPATCH_WINDOW=*time_window* ...

Description

Jobs from this queue are not dispatched outside of the dispatch window.

Default

Not defined. Dispatch window is always open.

DOCKER_IMAGE_AFFINITY

Syntax

DOCKER_IMAGE_AFFINITY=Y | y | N | n

Description

When scheduling Docker-based containerized jobs, setting this parameter to y or Y enables LSF to give preference for execution hosts that already have the requested Docker image. This reduces network bandwidth and the job start time because the execution host does not have to pull the Docker image from the repository and the job can immediately start on the execution host.

When this feature is enabled, LSF considers Docker image location information when scheduling Docker jobs. Docker image affinity interacts with host preference and **order[]** string requests in the following manner:

- If host preference is specified, the host preference is honored first. Among hosts with the same preference level, hosts with the requested Docker image are given higher priority.
- If the **order[]** string is specified, the hosts with the requested Docker image have a higher priority first. Among hosts that all have the requested Docker image, the **order[]** string is then honored.

The **CONTAINER** parameter must be defined for this parameter to work with this queue.

Default

Not defined.

ELIGIBLE_PEND_TIME_LIMIT

Specifies the eligible pending time limit for a job.

Syntax

ELIGIBLE_PEND_TIME_LIMIT=[*hour*:]*minute*

Description

LSF sends the queue-level eligible pending time limit configuration to IBM Spectrum LSF RTM (LSF RTM), which handles the alarm and triggered actions such as user notification (for example, notifying the user that submitted the job and the LSF administrator) and job control actions (for example, killing the job). LSF RTM compares the job's eligible pending time to the eligible pending time limit, and if the job is in an eligible pending state for longer than this specified time limit, LSF RTM triggers the alarm and actions. This parameter works without LSF RTM, but LSF does not take any other alarm actions.

In MultiCluster job forwarding mode, the job's eligible pending time limit is ignored in the execution cluster, while the submission cluster merges the job's queue-, application-, and job-level eligible pending time limit according to local settings.

The eligible pending time limit is in the form of [*hour*:]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The job-level eligible pending time limit (**bsub -ept1**) overrides the application-level limit (**ELIGIBLE_PEND_TIME_LIMIT** in `lsb.applications`), and the application-level limit overrides the queue-level limit specified here.

Default

Not defined.

ENABLE_GPU_HIST_RUN_TIME

Enables the use of historical GPU run time in the calculation of fairshare scheduling priority. Used only with fairshare scheduling.

Syntax

ENABLE_GPU_HIST_RUN_TIME=y | Y | n | N

Description

If set to Y, enables the use of historical GPU run time in the calculation of fairshare scheduling priority. If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Default

Not defined.

ENABLE_HIST_RUN_TIME

Enables the use of historical run time in the calculation of fairshare scheduling priority. Used only with fairshare scheduling.

Syntax

ENABLE_HIST_RUN_TIME=y | Y | n | N

Description

If set to Y, enables the use of historical run time in the calculation of fairshare scheduling priority. If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Default

Not defined.

ESTIMATED_RUNTIME**Syntax**

ESTIMATED_RUNTIME=[*hour*:]*minute*[/*host_name* | /*host_model*]

Description

This parameter specifies an estimated run time for jobs associated with a queue. LSF uses the ESTIMATED_RUNTIME value for scheduling purposes only, and does not kill jobs that exceed this value unless the jobs also exceed a defined RUNLIMIT. The format of runtime estimate is same as the RUNLIMIT parameter.

The job-level runtime estimate specified by **bsub -We** or the ESTIMATED_RUNTIME setting in an application override the ESTIMATED_RUNTIME setting in the queue. The ESTIMATED_RUNTIME setting in the queue overrides the cluster-wide ESTIMATED_RUNTIME setting.

The following LSF features use the ESTIMATED_RUNTIME value to schedule jobs:

- Job chunking
- Advance reservation
- SLA
- Slot reservation
- Backfill

- Allocation planner

Default

Not defined

EXCLUSIVE

Specifies an exclusive queue and compute unit type.

Syntax

EXCLUSIVE=Y | N | CU[*cu_type*]

Description

If set to Y, specifies an exclusive queue.

If set to CU, CU[], or CU[*cu_type*], specifies an exclusive queue as well as a queue exclusive to compute units of type *cu_type* (as defined in `lsb.params`). If no type is specified, the default compute unit type is used.

Jobs that are submitted to an exclusive queue with **bsub -x** are only dispatched to a host that has no other running jobs. Jobs that are submitted to a compute unit exclusive queue with **bsub -R "cu[excl]"** only run on a compute unit that has no other running jobs.

For hosts shared under the MultiCluster resource leasing model, jobs are not dispatched to a host that has running jobs, even if the jobs are from another cluster.

Note: EXCLUSIVE=Y or EXCLUSIVE=CU[*cu_type*] must be configured to enable affinity jobs to use CPUs exclusively, when the `all_jobs` scope is specified in the `exclusive` option of an `affinity[]` resource requirement string.

Default

N

EXEC_DRIVER

Syntax

EXEC_DRIVER=context[user(*user_name*)] starter[/*file_path_serverdir*/docker-starter.py]
controller[/*file_path_to_serverdir*/docker-control.py] monitor[/*file_path_to_serverdir*/
docker-monitor.py]

Replace *file_path_to_serverdir* with the actual file path of the `LSF_SERVERDIR` directory.

Description

Specifies the execution driver framework for Docker container jobs in this queue. This parameter uses the following keyword:

user

Optional. This keyword specifies the user account for starting scripts. The configured value is a user name instead of a user ID. For Docker jobs, this user must be a member of the Docker user group. By default this is the LSF primary administrator.

Note: This cannot be the root user.

LSF includes three execution driver scripts that are used to start a job (`docker-starter.py`), monitor the resource of a job (`docker-monitor.py`), and send a signal to a job (`docker-control.py`). These scripts are located in the `LSF_SERVERDIR` directory. Change the owner of the script files to the context user and change the file permissions to 700 or 500 before using them in the **EXEC_DRIVER** parameter.

Interaction with the **CONTAINER** parameter for Docker jobs

For Docker jobs, the **EXEC_DRIVER** parameter interacts with the following keywords in the **CONTAINER** parameter:

- **image**, which specifies the image name (**\$LSB_CONTAINER_IMAGE** environment variable) is supported when specifying the script names.
- **options** with runtime options and the option script is supported.

Example

```
Begin Queue
NAME = dockerq
CONTAINER = docker[image(repository.example.com:5000/file/path/ubuntu:latest)
             options(--rm --network=host --ipc=host -v /path/to/my/passwd:/etc/passwd)]
EXEC_DRIVER = context[user(user-name)] starter[/path/to/driver/docker-starter.py]
             controller[/path/to/driver/docker-control.py]
             monitor[/path/to/driver/docker-monitor.py]
DESCRIPTION = Docker User Service
End Queue
```

Default

Undefined

EXTENDABLE_RUNLIMIT

Enables the LSF allocation planner to extend the run limits of a job by changing its soft run limit if the resources that are occupied by this job are not needed by other jobs in queues with the same or higher priority. A soft run limit can be extended, while a hard run limit cannot be extended. The allocation planner looks at job plans of other jobs to determine if there are other jobs that require this job's resources.

Syntax

```
EXTENDABLE_RUNLIMIT=BASE[minutes] INCREMENT[minutes] GRACE[minutes] REQUEUE[Y|N]
```

Description

If configured, this parameter applies an extendable run limit policy to jobs with plans in the queue. When a job with a plan is dispatched, LSF sets an initial soft run limit for the job. Whenever a job reaches the soft run limit, LSF considers whether another job has a planned allocation on the resources. If not, LSF extends the job's soft run limit. Otherwise, LSF sets a hard run limit for the job. Whenever a job reaches the hard run limit, LSF terminates or requeues the job.

This parameter uses the following keywords:

BASE[minutes]

The initial soft run limit that is imposed on jobs in the queue. Whenever the job reaches the soft run limit, the allocation planner considers whether the resources that are held by the job are needed by another job in the queue by looking at plans for other jobs. If the resources are not required, LSF extends the soft run limit for the job. Otherwise, LSF sets a hard run limit.

Specify an integer value for the initial soft run limit.

INCREMENT[minutes]

If LSF decides to extend the soft run limit for the job, this keyword specifies the amount of time that LSF extends the soft run limit.

Specify an integer value for the soft run limit extension time. The default value is the value of the **BASE**[] keyword.

GRACE [*minutes*]

If LSF decides not to extend the soft run limit for the job, a hard run limit is set for this amount of minutes from the time the decision is made.

The default value is 0 (the job is terminated or requeued immediately).

REQUEUE [**Y** | **N**]

Specifies the action that LSF takes when a job reaches its hard run limit. If set to N, LSF terminates the job. If set to Y LSF requeues the job.

The default value is N (LSF terminates the job once the job reaches its hard run limit).

Default

Not defined.

Jobs that reach the specified run limit time (as specified by the **RUNLIMIT** parameter or the -W option) are checkpointed (if checkpointable), then terminated, regardless of whether resources are available.

FAIRSHARE

Enables queue-level user-based fairshare and specifies share assignments.

Syntax

```
FAIRSHARE=USER_SHARES[[user, number_shares] ...]
```

- Specify at least one user share assignment.
- Enclose the list in square brackets, as shown.
- Enclose each user share assignment in square brackets, as shown.
- *user*: specify users who are also configured to use queue. You can assign the shares to the following types of users:
 - A single user (specify *user_name*). To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).
 - Users in a group, individually (specify *group_name@*) or collectively (specify *group_name*). To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME\group_name*).
 - Users not included in any other share assignment, individually (specify the keyword *default*) or collectively (specify the keyword *others*)
 - By default, when resources are assigned collectively to a group, the group members compete for the resources on a first-come, first-served (FCFS) basis. You can use hierarchical fairshare to further divide the shares among the group members.
 - When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.
- *number_shares*
 - Specify a positive integer that represents the number of shares of the cluster resources that are assigned to the user.
 - The number of shares that are assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares that are assigned in each share assignment.

Description

Enables queue-level user-based fairshare and specifies optional share assignments. If share assignments are specified, only users with share assignments can submit jobs to the queue.

Compatibility

Do not configure hosts in a cluster to use fairshare at both queue and host levels. However, you can configure user-based fairshare and queue-based fairshare together.

Default

Not defined. No fairshare.

FAIRSHARE_ADJUSTMENT_FACTOR

Specifies the fairshare adjustment plug-in weighting factor. Used only with fairshare scheduling.

Syntax

FAIRSHARE_ADJUSTMENT_FACTOR=*number*

Description

In the calculation of a user dynamic share priority, this factor determines the relative importance of the user-defined adjustment that is made in the fairshare plug-in (`libfairshareadjust.*`).

A positive float number both enables the fairshare plug-in and acts as a weighting factor.

If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Default

Not defined.

FAIRSHARE_QUEUES

Defines cross-queue fairshare.

Syntax

FAIRSHARE_QUEUES=*queue_name*[*queue_name* ...]

Description

When this parameter is defined:

- The queue in which this parameter is defined becomes the “*master queue*”.
- Queues that are listed with this parameter are *subordinate queues* and inherit the fairshare policy of the master queue.
- A user has the same priority across the master and subordinate queues. If the same user submits several jobs to these queues, user priority is calculated by taking into account all the jobs that the user submitted across the master-subordinate set.

Notes

- By default, the PRIORITY range that is defined for queues in cross-queue fairshare cannot be used with any other queues. For example, you have 4 queues: `queue1`, `queue2`, `queue3`, `queue4`. You assign the following cross-queue fairshare: priorities
 - `queue1` priority: 30
 - `queue2` priority: 40
 - `queue3` priority: 50
- By default, the priority of `queue4` (which is not part of the cross-queue fairshare) cannot fall between the priority range of the cross-queue fairshare queues (30-50). It can be any number up to 29 or higher than 50. It does not matter if `queue4` is a fairshare queue or FCFS queue. If `DISPATCH_ORDER=QUEUE`

is set in the master queue, the priority of queue4 (which is not part of the cross-queue fairshare) can be any number, including a priority that falls between the priority range of the cross-queue fairshare queues (30-50).

- FAIRSHARE must be defined in the master queue. If it is also defined in the queues that are listed in FAIRSHARE_QUEUES, it is ignored.
- Cross-queue fairshare can be defined more than once within lsb.queues. You can define several sets of master-slave queues. However, a queue cannot belong to more than one master-slave set. For example, you can define:
 - In queue normal: FAIRSHARE_QUEUES=short
 - In queue priority: FAIRSHARE_QUEUES=night owners

Restriction: You cannot, however, define night, owners, or priority as subordinates in the queue normal; or normal and short as subordinates in the priority queue; or short, night, owners as master queues of their own.
- Cross-queue fairshare cannot be used with host partition fairshare. It is part of queue-level fairshare.
- Cross-queue fairshare cannot be used with absolute priority scheduling.

Default

Not defined

FILELIMIT

Specifies the per-process file size limit for all job processes from this queue.

Syntax

FILELIMIT=*integer*

Description

Set this parameter to place a per-process hard file size limit, in KB, for all of the processes that belong to a job from this queue (see **getrlimit(2)**).

Default

Unlimited

FWD_JOB_FACTOR

Forwarded job slots weighting factor. Used only with fairshare scheduling.

Syntax

FWD_JOB_FACTOR=*number*

Description

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the number of forwarded job slots reserved and in use by a user when using the LSF multicluster capability.

If undefined, the cluster-wide value from the lsb.params parameter of the same name is used.

Default

Not defined.

See also

RUN_JOB_FACTOR

FWD_USERS

Specifies a space-separated list of user names or user groups that can forward jobs to remote clusters in this queue when using the LSF multicluster capability.

Syntax

```
FWD_USERS=all [~user_name ...] [~user_group ...] | [user_name ...] [user_group [~user_group ...] ...]
```

Description

If user groups are specified, each user in the group can forward jobs to remote clusters in this queue. Use local user groups when specifying user groups.

User names must be valid login names. To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).

User group names can be LSF user groups or UNIX and Windows user groups. To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_group*).

Use the keyword `all` to specify all users or user groups in a cluster.

Use the not operator (`~`) to exclude users from the all specification or from user groups. This is useful if you have a large number of users but only want to exclude a few users or groups from the queue definition.

The not operator (`~`) can only be used with the `all` keyword or to exclude users from user groups.

Default

`all` (all users can forward jobs to remote clusters in the queue)

Examples

- `FWD_USERS=user1 user2`
- `FWD_USERS=all ~user1 ~user2`
- `FWD_USERS=all ~ugroup1`
- `FWD_USERS=groupA ~user3 ~user4`

GPU_REQ

Specify GPU requirements together in one statement.

Syntax

```
you GPU_REQ = "[num=num_gpus[/task | host]] [:mode=shared | exclusive_process]
[:mps=yes[, shared] | no | per_socket[, shared] | per_gpu[, shared]] [:j_exclusive=yes | no]
[:gmodel=model_name[#mem_size]] [:gtile=tile_num|'!' ] [:gmem=mem_value] [:nvlink=yes]
[:aff=yes | no]"
```

Description

The **LSB_GPU_NEW_SYNTAX=extend** or **LSB_GPU_NEW_SYNTAX=Y** parameter must be set in the `lsf.conf` file to specify the `j_exclusive`, `mode`, `mps`, and `num` GPU requirements in the **GPU_REQ** parameter.

The **LSB_GPU_NEW_SYNTAX=extend** must be set in the `lsf.conf` file to specify the `gmem`, `gmodel`, `gtile`, and `nvlink` GPU requirements.

When the **LSB_GPU_NEW_SYNTAX=extend** parameter is set, the resource requirements of your job cannot use the legacy GPU resources (`ngpus_shared`, `ngpus_excl_t`, `ngpus_excl_p`) as job resource requirements.

The **GPU_REQ** parameter takes the following arguments:

num=num_gpus[/task | host]

The number of physical GPUs required by the job. By default, the number is per host. You can also specify that the number is per task by specifying /task after the number.

If you specified that the number is per task, the configuration of the `ngpus_physical` resource in the `lsb.resources` file is set to `PER_TASK`, or the **RESOURCE_RESERVE_PER_TASK=Y** parameter is set in the `lsb.params` file, this number is the requested count per task.

mode=shared | exclusive_process

The GPU mode when the job is running, either `shared` or `exclusive_process`. The default mode is `shared`.

The `shared` mode corresponds to the NVIDIA DEFAULT compute mode. The `exclusive_process` mode corresponds to the NVIDIA EXCLUSIVE_PROCESS compute mode.

mps=yes[,shared] | per_socket[,shared] | per_gpu[,shared] | no

Enables or disables the NVIDIA Multi-Process Service (MPS) for the GPUs that are allocated to the job. Using MPS effectively causes the EXCLUSIVE_PROCESS mode to behave like the DEFAULT mode for all MPS clients. MPS always allows multiple clients to use the GPU through the MPS server.

MPS is useful for both shared and exclusive process GPUs, and allows more efficient sharing of GPU resources and better GPU utilization. See the NVIDIA documentation for more information and limitations.

When using MPS, use the EXCLUSIVE_PROCESS mode to ensure that only a single MPS server is using the GPU, which provides additional insurance that the MPS server is the single point of arbitration between all CUDA process for that GPU.

You can also enable MPS daemon sharing by adding the **share** keyword with a comma and no space (for example, `mps=yes , shared` enables MPS daemon sharing on the host). If sharing is enabled, all jobs that are submitted by the same user with the same resource requirements share the same MPS daemon on the host, socket, or GPU.

LSF starts MPS daemons on a per-host, per-socket, or per-GPU basis depending on value of the **mps** keyword:

- If `mps=yes` is set, LSF starts one MPS daemon per host per job.
 - When `share` is enabled (that is, if `mps=yes , shared` is set), LSF starts one MPS daemon per host for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon on the host.
- If `mps=per_socket` is set, LSF starts one MPS daemon per socket per job. When enabled with `share` (that is, if `mps=per_socket , shared` is set), LSF starts one MPS daemon per socket for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon for the socket.
- If `mps=per_gpu` is set, LSF starts one MPS daemon per GPU per job. When enabled with `share` (that is, if `mps=per_gpu , shared` is set), LSF starts one MPS daemon per GPU for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon for the GPU.

Note: Using EXCLUSIVE_THREAD mode with MPS is not supported and might cause unexpected behavior.

j_exclusive=yes | no

Specifies whether the allocated GPUs can be used by other jobs. When the mode is set to `exclusive_process`, the `j_exclusive=yes` option is set automatically.

aff=yes | no

Specifies whether to enforce strict GPU-CPU affinity binding. If set to `no`, LSF relaxes GPU affinity while maintaining CPU affinity. By default, `aff=yes` is set to maintain strict GPU-CPU affinity binding.

The **LSB_GPU_NEW_SYNTAX=extend** or **LSB_GPU_NEW_SYNTAX=Y** parameter must be defined in the `lsf.conf` file to enable the `-gpu` option and **GPU_REQ** parameter syntax. When the

LSB_GPU_NEW_SYNTAX=extend parameter is set, you cannot use the GPU resources `ngpus_shared`, `ngpus_excl_t` and `ngpus_excl_p` as job resource requirements.

If the **GPU_REQ_MERGE** parameter is defined as Y or y in the `lsb.params` file and a GPU requirement is specified at multiple levels (at least two of the default cluster, queue, application profile, or job level requirements), each option of the GPU requirement is merged separately. Job level overrides application level, which overrides queue level, which overrides the default cluster GPU requirement. For example, if the mode option of the GPU requirement is defined on the `-gpu` option, and the mps option is defined in the queue, the mode of job level and the mps value of queue is used.

If the **GPU_REQ_MERGE** parameter is not defined as Y or y in the `lsb.params` file and a GPU requirement is specified at multiple levels (at least two of the default cluster, queue, application profile, or job level requirements), the entire GPU requirement string is replaced. The entire job level GPU requirement string overrides application level, which overrides queue level, which overrides the default GPU requirement.

The **esub** parameter **LSB_SUB4_GPU_REQ** modifies the value of the `-gpu` option.

LSF selects the GPU that meets the topology requirement first. If the GPU mode of the selected GPU is not the requested mode, LSF changes the GPU to the requested mode. For example, if LSF allocates an `exclusive_process` GPU to a job that needs a shared GPU, LSF changes the GPU mode to shared before the job starts and then changes the mode back to `exclusive_process` when the job finishes.

The GPU requirements are converted to `rusage` resource requirements for the job. For example, `num=2` is converted to `rusage[ngpus_physical=2]`. Use the **bjobs**, **bhist**, and **bacct** commands to see the merged resource requirement.

There might be complex GPU requirements that the **bsub -gpu** option and **GPU_REQ** parameter syntax cannot cover, including compound GPU requirements (for different GPU requirements for jobs on different hosts, or for different parts of a parallel job) and alternate GPU requirements (if more than one set of GPU requirements might be acceptable for a job to run). For complex GPU requirements, use the **bsub -R** command option, or the **RES_REQ** parameter in the `lsb.applications` or `lsb.queues` file to define the resource requirement string.

Note: You can define the mode, `j_exclusive`, and mps options only with the `-gpu` option, the **LSB_GPU_REQ** parameter in the `lsf.conf` file, or the **GPU_REQ** parameter in the `lsb.queues` or `lsb.applications` files. You cannot use these options with the `rusage` resource requirement string in the **bsub -R** command option or the **RES_REQ** parameter in the `lsb.queues` or `lsb.applications` files.

Default

Not defined

See also

- **LSB_GPU_REQ**
- **LSB_GPU_NEW_SYNTAX**
- **bsub -gpu**

GPU_RUN_TIME_FACTOR

Specifies the GPU run time weighting factor. Used only with fairshare scheduling.

Syntax

`GPU_RUN_TIME_FACTOR=number`

Description

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the total GPU run time of a user's running GPU jobs.

If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Default

Not defined.

HIST_HOURS

Determines a rate of decay for cumulative CPU time, run time, and historical run time. Used only with fairshare scheduling.

Syntax

`HIST_HOURS=hours`

Description

To calculate dynamic user priority, LSF uses a decay factor to scale the actual CPU time and run time. One hour of recently used time is equivalent to 0.1 hours after the specified number of hours elapses.

To calculate dynamic user priority with decayed run time and historical run time, LSF uses the same decay factor to scale the accumulated run time of finished jobs and run time of running jobs, so that one hour of recently used time is equivalent to 0.1 hours after the specified number of hours elapses.

When `HIST_HOURS=0`, CPU time and run time that is accumulated by running jobs is not decayed.

If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Default

Not defined.

HJOB_LIMIT

Specifies the per-host job slot limit.

Syntax

`HJOB_LIMIT=integer`

Description

This parameter defines the maximum number of job slots that this queue can use on any host. This limit is configured per host, regardless of the number of processors it might have.

Note: When `LSB_ENABLE_HPC_ALLOCATION=Y` is defined, all slots on the execution hosts must be allocated to exclusive jobs. If the **HJOB_LIMIT** value for the queue is less than the maximum number of slots on the execution host, jobs submitted to this queue remaining pending indefinitely. This is because LSF cannot allocate all slots in this host for your exclusive job.

Example

The following queue runs a maximum of one job on each of `hostA`, `hostB`, and `hostC`:

```
Begin Queue
...
HJOB_LIMIT = 1
HOSTS=hostA hostB hostC
...
End Queue
```

Default

Unlimited

HOST_POST_EXEC

Enables host-based post-execution processing at the queue level.

Syntax

`HOST_POST_EXEC=command`

Description

The **HOST_POST_EXEC** command runs on all execution hosts after the job finishes. If job-based post-execution **POST_EXEC** was defined at the queue-level, application-level, or job-level, the **HOST_POST_EXEC** command runs after **POST_EXEC** of any level.

Host-based post-execution commands can be configured at the queue and application level, and run in the following order:

1. The application-level command
2. The queue-level command.

The supported command rule is the same as the existing **POST_EXEC** for the queue section. See the **POST_EXEC** topic for details.

Note: The host-based post-execution command cannot run on Windows systems. This parameter cannot be used to configure job-based post-execution processing.

Default

Not defined.

HOST_PRE_EXEC

Enables host-based pre-execution processing at the queue level.

Syntax

`HOST_PRE_EXEC=command`

Description

The **HOST_PRE_EXEC** command runs on all execution hosts before the job starts. If job based pre-execution **PRE_EXEC** was defined at the queue-level/application-level/job-level, the **HOST_PRE_EXEC** command runs before **PRE_EXEC** of any level.

Host-based pre-execution commands can be configured at the queue and application level, and run in the following order:

1. The queue-level command
2. The application-level command.

The supported command rule is the same as the existing **PRE_EXEC** for the queue section. See the **PRE_EXEC** topic for details.

Note: The host-based pre-execution command cannot be executed on Windows platforms. This parameter cannot be used to configure job-based pre-execution processing.

Default

Not defined.

HOSTLIMIT_PER_JOB

Specifies the per-job host limit.

Syntax

HOSTLIMIT_PER_JOB=*integer*

Description

This parameter defines the maximum number of hosts that a job in this queue can use. LSF verifies the host limit during the allocation phase of scheduling. If the number of hosts requested for a parallel job exceeds this limit and LSF cannot satisfy the minimum number of request slots, the parallel job will pend. However, for resumed parallel jobs, this parameter does not stop the job from resuming even if the job's host allocation exceeds the per-job host limit specified in this parameter.

Default

Unlimited

HOSTS

Specifies a space-separated list of hosts on which jobs from this queue can be run.

Syntax

HOSTS=*host_list* | none

- *host_list* is a space-separated list of the following items:
 - *host_name*[@*cluster_name*][[!] | +*pref_level*]
 - *host_partition*[+*pref_level*]
 - *host_group*[[!] | +*pref_level*]
 - *compute_unit*[[!] | +*pref_level*]
 - [~]*host_name*
 - [~]*host_group*
 - [~]*compute_unit*
- The list can include the following items only once:
 - all@*cluster_name*
 - others[+*pref_level*]
 - all
 - allremote
- The none keyword is only used with the MultiCluster job forwarding model, to specify a remote-only queue.

Description

If compute units, host groups, or host partitions are included in the list, the job can run on any host in the unit, group, or partition. All the members of the host list should either belong to a single host partition or not belong to any host partition. Otherwise, job scheduling may be affected.

Some items can be followed by a plus sign (+) and a positive number to indicate the preference for dispatching a job to that host. A higher number indicates a higher preference. If a host preference is not given, it is assumed to be 0. If there are multiple candidate hosts, LSF dispatches the job to the host with the highest preference; hosts at the same level of preference are ordered by load.

If compute units, host groups, or host partitions are assigned a preference, each host in the unit, group, or partition has the same preference.

Use the keyword `others` to include all hosts not explicitly listed.

Use the keyword `all` to include all hosts not explicitly excluded.

Use the keyword `all@cluster_name hostgroup_name` or `allremote hostgroup_name` to include lease in hosts.

Use the not operator (`~`) to exclude hosts from the `all` specification in the queue. This is useful if you have a large cluster but only want to exclude a few hosts from the queue definition.

The not operator can only be used with the `all` keyword. It is *not* valid with the keywords `others` and `none`.

The not operator (`~`) can be used to exclude host groups.

For parallel jobs, specify first execution host candidates when you want to ensure that a host has the required resources or runtime environment to handle processes that run on the first execution host.

To specify one or more hosts, host groups, or compute units as first execution host candidates, add the exclamation point (!) symbol after the name.

Follow these guidelines when you specify first execution host candidates:

- If you specify a compute unit or host group, you must first define the unit or group in the file `lsb.hosts`.
- Do not specify a dynamic host group as a first execution host.
- Do not specify `all`, `allremote`, or `others`, or a host partition as a first execution host.
- Do not specify a preference (+) for a host identified by (!) as a first execution host candidate.
- For each parallel job, specify enough regular hosts to satisfy the CPU requirement for the job. Once LSF selects a first execution host for the current job, the other first execution host candidates
 - Become unavailable to the current job
 - Remain available to other jobs as either regular or first execution hosts
- You cannot specify first execution host candidates when you use the `brun` command.

Restriction: If you have enabled EGO, host groups and compute units are not honored.

With MultiCluster resource leasing model, use the format `host_name@cluster_name` to specify a borrowed host. LSF does not validate the names of remote hosts. The keyword `others` indicates all local hosts not explicitly listed. The keyword `all` indicates all local hosts not explicitly excluded. Use the keyword `allremote` to specify all hosts borrowed from all remote clusters. Use `all@cluster_name` to specify the group of all hosts borrowed from one remote cluster. You cannot specify a host group or partition that includes remote resources, unless it uses the keyword `allremote` to include all remote hosts. You cannot specify a compute unit that includes remote resources.

With MultiCluster resource leasing model, the not operator (`~`) can be used to exclude local hosts or host groups. You cannot use the not operator (`~`) with remote hosts.

Restriction: Hosts that participate in queue-based fairshare cannot be in a host partition.

Behavior with host intersection

Host preferences specified by `bsub -m` combine intelligently with the queue specification and advance reservation hosts. The jobs run on the hosts that are both specified at job submission and belong to the queue or have advance reservation.

Example 1

```
HOSTS=hostA+1 hostB hostC+1 hostD+3
```

This example defines three levels of preferences: run jobs on hostD as much as possible, otherwise run on either hostA or hostC if possible, otherwise run on hostB. Jobs should not run on hostB unless all other hosts are too busy to accept more jobs.

Example 2

```
HOSTS=hostD+1 others
```

Run jobs on hostD as much as possible, otherwise run jobs on the least-loaded host available.

With MultiCluster resource leasing model, this queue does not use borrowed hosts.

Example 3

```
HOSTS=all ~hostA
```

Run jobs on all hosts in the cluster, except for hostA.

With MultiCluster resource leasing model, this queue does not use borrowed hosts.

Example 4

```
HOSTS=Group1 ~hostA hostB hostC
```

Run jobs on hostB, hostC, and all hosts in Group1 except for hostA.

With MultiCluster resource leasing model, this queue uses borrowed hosts if Group1 uses the keyword allremote.

Example 5

```
HOSTS=hostA! hostB+ hostC hostgroup1!
```

Runs parallel jobs using either hostA or a host defined in hostgroup1 as the first execution host. If the first execution host cannot run the entire job due to resource requirements, runs the rest of the job on hostB. If hostB is too busy to accept the job, or if hostB does not have enough resources to run the entire job, runs the rest of the job on hostC.

Example 6

```
HOSTS=computeunit1! hostB hostC
```

Runs parallel jobs using a host in computeunit1 as the first execution host. If the first execution host cannot run the entire job due to resource requirements, runs the rest of the job on other hosts in computeunit1 followed by hostB and finally hostC.

Example 7

```
HOSTS=hostgroup1! computeunitA computeunitB computeunitC
```

Runs parallel jobs using a host in hostgroup1 as the first execution host. If additional hosts are required, runs the rest of the job on other hosts in the same compute unit as the first execution host, followed by hosts in the remaining compute units in the order they are defined in the lsb.hosts ComputeUnit section.

Default

all (the queue can use all hosts in the cluster, and every host has equal preference)

With MultiCluster resource leasing model, this queue can use all local hosts, but no borrowed hosts.

IGNORE_DEADLINE

Disables deadline constraint scheduling.

Syntax

IGNORE_DEADLINE=Y

Description

If set to Y, LSF ignores deadline constraint scheduling and starts all jobs regardless of deadline constraints.

IMPT_JOBKLG

Specifies the MultiCluster pending job limit for a receive-jobs queue. Used only with the MultiCluster job forwarding model.

Syntax

IMPT_JOBKLG=*integer* |infinite

Description

This parameter represents the maximum number of MultiCluster jobs that can be pending in the queue; once the limit has been reached, the queue stops accepting jobs from remote clusters.

Use the keyword infinite to make the queue accept an unlimited number of pending MultiCluster jobs.

Default

50

IMPT_TASKKLG

Specifies the MultiCluster pending job task limit for a receive-jobs queue. Used only with the MultiCluster job forwarding model.

Syntax

IMPT_TASKKLG=*integer* |infinite

Description

In the submission cluster, if the total of requested job tasks and the number of imported pending tasks in the receiving queue is greater than **IMPT_TASKKLG**, the queue stops accepting jobs from remote clusters, and the job is not forwarded to the receiving queue.

Specify an integer between 0 and 2147483646 for the number of tasks.

Use the keyword infinite to make the queue accept an unlimited number of pending MultiCluster job tasks.

Set **IMPT_TASKKLG** to 0 to forbid any job being forwarded to the receiving queue.

Note: **IMPT_SLOTKLG** has been changed to **IMPT_TASKKLG** and the concept has changed from slot to task as of LSF 9.1.3,

Default

infinite (The queue accepts an unlimited number of pending MultiCluster job tasks.)

INTERACTIVE

Specifies whether the queue accepts interactive or non-interactive jobs.

Syntax

INTERACTIVE=YES | NO | ONLY

Description

If set to YES, causes the queue to accept both interactive and non-interactive batch jobs.

If set to NO, causes the queue to reject interactive batch jobs.

If set to ONLY, causes the queue to accept interactive batch jobs and reject non-interactive batch jobs.

Interactive batch jobs are submitted via **bsub -I**.

Default

YES. The queue accepts both interactive and non-interactive jobs.

INTERRUPTIBLE_BACKFILL

Configures interruptible backfill scheduling policy, which allows reserved job slots to be used by low priority small jobs that are terminated when the higher priority large jobs are about to start.

Syntax

INTERRUPTIBLE_BACKFILL=*seconds*

Description

There can only be one interruptible backfill queue. It should be the lowest priority queue in the cluster.

Specify the minimum number of seconds for the job to be considered for backfilling. This minimal time slice depends on the specific job properties; it must be longer than at least one useful iteration of the job. Multiple queues may be created if a site has jobs of distinctively different classes.

An interruptible backfill job:

- Starts as a regular job and is killed when it exceeds the queue runtime limit, or
- Is started for backfill whenever there is a backfill time slice longer than the specified minimal time, and killed before the slot-reservation job is about to start

The queue RUNLIMIT corresponds to a maximum time slice for backfill, and should be configured so that the wait period for the new jobs submitted to the queue is acceptable to users. 10 minutes of runtime is a common value.

You should configure REQUEUE_EXIT_VALUES for interruptible backfill queues.

BACKFILL and RUNLIMIT must be configured in the queue. The queue is disabled if BACKFILL and RUNLIMIT are not configured.

Assumptions and limitations:

- The interruptible backfill job holds the slot-reserving job start until its calculated start time, in the same way as a regular backfill job. The interruptible backfill job are not preempted in any way other than being killed when its time come.
- While the queue is checked for the consistency of interruptible backfill, backfill and runtime specifications, the requeue exit value clause is not verified, nor executed automatically. Configure requeue exit values according to your site policies.
- The interruptible backfill job must be able to do at least one unit of useful calculations and save its data within the minimal time slice, and be able to continue its calculations after it has been restarted

- Interruptible backfill paradigm does not explicitly prohibit running parallel jobs, distributed across multiple nodes; however, the chance of success of such job is close to zero.

Default

Not defined. No interruptible backfilling.

JOB_ACCEPT_INTERVAL

Specifies the amount of time to wait after dispatching a job to a host before dispatching a second job to the same host.

Syntax

`JOB_ACCEPT_INTERVAL=integer`

Description

This parameter value is multiplied by the value of `lsb.params MBD_SLEEP_TIME` (60 seconds by default). The result of the calculation is the number of seconds to wait after dispatching a job to a host before dispatching a second job to the same host.

If set to 0 (zero), a host may accept more than one job in each dispatch turn. By default, there is no limit to the total number of jobs that can run on a host, so if this parameter is set to 0, a very large number of jobs might be dispatched to a host all at once. This can overload your system to the point that it is unable to create any more processes. It is not recommended to set this parameter to 0.

`JOB_ACCEPT_INTERVAL` set at the queue level (`lsb.queues`) overrides `JOB_ACCEPT_INTERVAL` set at the cluster level (`lsb.params`).

Note:

The parameter `JOB_ACCEPT_INTERVAL` only applies when there are running jobs on a host. A host running a short job which finishes before `JOB_ACCEPT_INTERVAL` has elapsed is free to accept a new job without waiting.

Default

Not defined. The queue uses `JOB_ACCEPT_INTERVAL` defined in `lsb.params`, which has a default value of 1.

JOB_ACTION_WARNING_TIME

Specifies the amount of time before a job control action occurs that a job warning action is to be taken.

Syntax

`JOB_ACTION_WARNING_TIME=[hour:]minute`

Description

Job action warning time is not normalized.

A job action warning time must be specified with a job warning action (the `JOB_WARNING_ACTION` parameter) in order for job warning to take effect.

The warning time specified by the `bsub -wt` option overrides `JOB_ACTION_WARNING_TIME` in the queue. `JOB_ACTION_WARNING_TIME` is used as the default when no command line option is specified.

Example

```
JOB_ACTION_WARNING_TIME=2
JOB_WARNING_ACTION=URG
```

2 minutes before the job reaches runtime limit or termination deadline, or the queue's run window is closed, an URG signal is sent to the job.

Default

Not defined

JOB_CONTROLS

Changes the behavior of the SUSPEND, RESUME, and TERMINATE actions in LSF.

Syntax

```
JOB_CONTROLS=SUSPEND[signal | command | CHPNT] RESUME[signal | command]
```

```
TERMINATE[signal | command | CHPNT]
```

- *signal* is a UNIX signal name (for example, SIGTSTP or SIGTERM). The specified signal is sent to the job. The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the **kill -l** command.
- *command* specifies a `/bin/sh` command line to be invoked.

Restriction:

Do not quote the command line inside an action definition. Do not specify a signal followed by an action that triggers the same signal. For example, do not specify `JOB_CONTROLS=TERMINATE[bkill]` or `JOB_CONTROLS=TERMINATE[brequeue]`. This causes a deadlock between the signal and the action.

- CHPNT is a special action, which causes the system to checkpoint the job. Only valid for SUSPEND and TERMINATE actions:
 - If the SUSPEND action is CHPNT, the job is checkpointed and then stopped by sending the SIGSTOP signal to the job automatically.
 - If the TERMINATE action is CHPNT, then the job is checkpointed and killed automatically.

Description

- The contents of the configuration line for the action are run with `/bin/sh -c` so you can use shell features in the command.
- The standard input, output, and error of the command are redirected to the NULL device, so you cannot tell directly whether the command runs correctly. The default null device on UNIX is `/dev/null`.
- The command is run as the user of the job.
- All environment variables set for the job are also set for the command action. The following additional environment variables are set:
 - LSB_JOBPGIDS: a list of current process group IDs of the job
 - LSB_JOBPIIDS: a list of current process IDs of the job
- For the SUSPEND action command, the following environment variables are also set:
 - LSB_SUSP_REASONS - an integer representing a bitmap of suspending reasons as defined in `lsbatch.h`. The suspending reason can allow the command to take different actions based on the reason for suspending the job.
 - LSB_SUSP_SUBREASONS - an integer representing the load index that caused the job to be suspended. When the suspending reason `SUSP_LOAD_REASON` (suspended by load) is set in `LSB_SUSP_REASONS`, `LSB_SUSP_SUBREASONS` set to one of the load index values defined in `lsf.h`. Use `LSB_SUSP_REASONS` and `LSB_SUSP_SUBREASONS` together in your custom job control to determine the exact load threshold that caused a job to be suspended.
- If an additional action is necessary for the SUSPEND command, that action should also send the appropriate signal to the application. Otherwise, a job can continue to run even after being suspended by LSF. For example, `JOB_CONTROLS=SUSPEND[kill $LSB_JOBPIIDS; command]`

- If the job control command fails, LSF retains the original job status.
- If you set preemption with the signal SIGTSTP you use IBM Spectrum LSF License Scheduler, define **LIC_SCHED_PREEMPT_STOP=Y** in `lsf.conf` for License Scheduler preemption to work.

Note: When you use **b1launch** to run parallel jobs on multiple hosts, job control actions defined in **JOB_CONTROLS** in `lsb.queues` only take effect on the first execution host. Job control actions defined in the queue do not affect tasks running on other hosts. If **JOB_CONTROLS** is defined, the default job control signals of LSF (SUSPEND, RESUME, TERMINATE) do not reach each task on each execution host.

Default

On UNIX, by default, SUSPEND sends SIGTSTP for parallel or interactive jobs and SIGSTOP for other jobs. RESUME sends SIGCONT. TERMINATE sends SIGINT, SIGTERM and SIGKILL in that order.

On Windows, actions equivalent to the UNIX signals have been implemented to do the default job control actions. Job control messages replace the SIGINT and SIGTERM signals, but only customized applications are able to process them. Termination is implemented by the `TerminateProcess()` system call.

JOB_IDLE

Specifies a threshold for idle job exception handling.

Syntax

`JOB_IDLE=number`

Description

The value should be a number between 0.0 and 1.0 representing CPU time/runtime. If the job idle factor is less than the specified threshold, LSF invokes `LSF_SERVERDIR/eadmin` to trigger the action for a job idle exception.

The minimum job run time before `mbatchd` reports that the job is idle is defined as `DETECT_IDLE_JOB_AFTER` in `lsb.params`.

Valid values

Any positive number between 0.0 and 1.0

Example

```
JOB_IDLE=0.10
```

A job idle exception is triggered for jobs with an idle value (CPU time/runtime) less than 0.10.

Default

Not defined. No job idle exceptions are detected.

JOB_OVERRUN

Specifies a threshold for job overrun exception handling.

Syntax

`JOB_OVERRUN=run_time`

Description

If a job runs longer than the specified run time, LSF invokes `LSF_SERVERDIR/eadmin` to trigger the action for a job overrun exception.

Example

```
JOB_OVERRUN=5
```

A job overrun exception is triggered for jobs running longer than 5 minutes.

Default

Not defined. No job overrun exceptions are detected.

JOB_SIZE_LIST

Specifies a list of job sizes (number of tasks) that are allowed on this queue.

Syntax

```
JOB_SIZE_LIST=default_size [size ...]
```

Description

When submitting a job or modifying a pending job that requests a job size by using the `-n` or `-R` options for **bsub** and **bmod**, the requested job size must be a single fixed value that matches one of the values that **JOB_SIZE_LIST** specifies, which are the job sizes that are allowed on this queue. LSF rejects the job if the requested job size is not in this list. In addition, when using **bswitch** to switch a pending job with a requested job size to another queue, the requested job size in the pending job must also match one of the values in **JOB_SIZE_LIST** for the new queue.

The first value in this list is the default job size, which is the assigned job size request if the job was submitted without requesting one. The remaining values are the other job sizes allowed in the queue, and may be defined in any order.

When defined in both a queue and an application profile (`lsb.applications`), the job size request must satisfy both requirements. In addition, **JOB_SIZE_LIST** overrides any **TASKLIMIT** parameters defined at the same level. Job size requirements do not apply to queues and application profiles with no job size lists, nor do they apply to other levels of job submissions (that is, host level or cluster level job submissions).

Note: An exclusive job may allocate more slots on the host than is required by the tasks. For example, if `JOB_SIZE_LIST=8` and an exclusive job requesting `-n8` runs on a 16 slot host, all 16 slots are assigned to the job. The job runs as expected, since the 8 tasks specified for the job matches the job size list.

Valid values

A space-separated list of positive integers between 1 and 2147483646.

Default

Undefined

JOB_STARTER

Creates a specific environment for submitted jobs prior to execution.

Syntax

```
JOB_STARTER=starter [starter] ["%USRCMD"] [starter]
```

Description

starter is any executable that can be used to start the job (i.e., can accept the job as an input argument). Optionally, additional strings can be specified.

By default, the user commands run after the job starter. A special string, %USRCMD, can be used to represent the position of the user's job in the job starter command line. The %USRCMD string and any additional commands must be enclosed in quotation marks (" ").

If your job starter script runs on a Windows execution host and includes symbols (like & or |), you can use the **JOB_STARTER_EXTEND=preservestarter** parameter in `lsf.conf` and set **JOB_STARTER=preservestarter** in `lsb.queues`. A customized **userstarter** can also be used.

Example

```
JOB_STARTER=csh -c "%USRCMD;sleep 10"
```

In this case, if a user submits a job

```
% bsub myjob arguments
```

the command that actually runs is:

```
% csh -c "myjob arguments;sleep 10"
```

Default

Not defined. No job starter is used.

JOB_UNDERRUN

Specifies a threshold for job underrun exception handling.

Syntax

```
JOB_UNDERRUN=run_time
```

Description

If a job exits before the specified number of minutes, LSF invokes `LSF_SERVERDIR/eadmin` to trigger the action for a job underrun exception.

Example

```
JOB_UNDERRUN=2
```

A job underrun exception is triggered for jobs running less than 2 minutes.

Default

Not defined. No job underrun exceptions are detected.

JOB_WARNING_ACTION

Specifies the job action to be taken before a job control action occurs.

Syntax

```
JOB_WARNING_ACTION=signal
```

Description

A job warning action must be specified with a job action warning time (the **JOB_ACTION_WARNING_TIME** parameter) in order for job warning to take effect.

If **JOB_WARNING_ACTION** is specified, LSF sends the warning action to the job before the actual control action is taken. This allows the job time to save its result before being terminated by the job control action.

The warning action specified by the **bsub -wa** option overrides **JOB_WARNING_ACTION** in the queue. **JOB_WARNING_ACTION** is used as the default when no command line option is specified.

Example

```
JOB_ACTION_WARNING_TIME=2
JOB_WARNING_ACTION=URG
```

2 minutes before the job reaches runtime limit or termination deadline, or the queue's run window is closed, an URG signal is sent to the job.

Default

Not defined

LOAD_INDEX

Specifies scheduling and suspending thresholds for the specified dynamic load index.

Syntax

```
load_index=loadSched[/loadStop]
```

Specify *io*, *it*, *ls*, *mem*, *pg*, *r15s*, *r1m*, *r15m*, *swp*, *tmp*, *ut*, or a non-shared custom external load index. Specify multiple lines to configure thresholds for multiple load indices.

Specify *io*, *it*, *ls*, *mem*, *pg*, *r15s*, *r1m*, *r15m*, *swp*, *tmp*, *ut*, or a non-shared custom external load index as a column. Specify multiple columns to configure thresholds for multiple load indices.

Description

The *loadSched* condition must be satisfied before a job is dispatched to the host. If a *RESUME_COND* is not specified, the *loadSched* condition must also be satisfied before a suspended job can be resumed.

If the *loadStop* condition is satisfied, a job on the host is suspended.

The *loadSched* and *loadStop* thresholds permit the specification of conditions using simple AND/OR logic. Any load index that does not have a configured threshold has no effect on job scheduling.

LSF does not suspend a job if the job is the only batch job running on the host and the machine is interactively idle (*it*>0).

The *r15s*, *r1m*, and *r15m* CPU run queue length conditions are compared to the effective queue length as reported by **lsload -E**, which is normalized for multiprocessor hosts. Thresholds for these parameters should be set at appropriate levels for single processor hosts.

Example

```
MEM=100/10
SWAP=200/30
```

These two lines translate into a *loadSched* condition of

```
mem>=100 && swap>=200
```

and a *loadStop* condition of

```
mem < 10 || swap < 30
```

Default

Not defined

LOCAL_MAX_PREEEXEC_RETRY

Specifies the maximum number of times to attempt the pre-execution command of a job on the local cluster.

Syntax

LOCAL_MAX_PREEEXEC_RETRY=*integer*

Description

When this limit is reached, the default behavior of the job is defined by the **LOCAL_MAX_PREEEXEC_RETRY_ACTION** parameter in `lsb.params`, `lsb.queues`, or `lsb.applications`.

Valid values

$0 < \text{MAX_PREEEXEC_RETRY} < \text{INFINIT_INT}$

INFINIT_INT is defined in `lsf.h`.

Default

Not defined. The number of preexec retry times is unlimited

See also

LOCAL_MAX_PREEEXEC_RETRY_ACTION in `lsb.params`, `lsb.queues`, and `lsb.applications`.

LOCAL_MAX_PREEEXEC_RETRY_ACTION

The default behavior of a job when it reaches the maximum number of times to attempt its pre-execution command on the local cluster.

Syntax

LOCAL_MAX_PREEEXEC_RETRY_ACTION=SUSPEND | EXIT

Description

This parameter specifies the default behavior of a job when it reaches the maximum number of times to attempt its pre-execution command on the local cluster (as specified by **LOCAL_MAX_PREEEXEC_RETRY** in `lsb.params`, `lsb.queues`, or `lsb.applications`).

- If set to SUSPEND, the local or leased job is suspended and its status is set to PSUSP
- If set to EXIT, the local or leased job exits and its status is set to EXIT. The job exits with the same exit code as the last pre-execution fail exit code.

This parameter is configured cluster-wide (`lsb.params`), at the queue level (`lsb.queues`), and at the application level (`lsb.applications`). The action specified in `lsb.applications` overrides `lsb.queues`, and `lsb.queues` overrides the `lsb.params` configuration.

Default

Not defined. If not defined in `lsb.params`, the default action is SUSPEND.

See also

LOCAL_MAX_PREEEXEC_RETRY in `lsb.params`, `lsb.queues`, and `lsb.applications`.

MANDATORY_EXTSCHED

Specifies mandatory external scheduling options for the queue.

Syntax

MANDATORY_EXTSCHED=*external_scheduler_options*

Description

-extsched options on the **bsub** command are merged with MANDATORY_EXTSCHED options, and MANDATORY_EXTSCHED options override any conflicting job-level options set by -extsched.

Default

Not defined

MAX_JOB_PREEMPT

Specifies the maximum number of times a job can be preempted. Applies to queue-based preemption only.

Syntax

MAX_JOB_PREEMPT=*integer*

Valid values

0 < MAX_JOB_PREEMPT < INFINIT_INT

INFINIT_INT is defined in `lsf.h`.

Default

Not defined. The number of preemption times is unlimited.

MAX_JOB_REQUEUE

Specifies the maximum number of times to requeue a job automatically.

Syntax

MAX_JOB_REQUEUE=*integer*

Valid values

0 < MAX_JOB_REQUEUE < INFINIT_INT

INFINIT_INT is defined in `lsf.h`.

Default

Not defined. The number of requeue times is unlimited

MAX_PREEEXEC_RETRY

Use REMOTE_MAX_PREEEXEC_RETRY instead. This parameter is maintained for backwards compatibility. The maximum number of times to attempt the pre-execution command of a job from a remote cluster. Used only with the MultiCluster job forwarding model.

Syntax

MAX_PREEEXEC_RETRY=*integer*

Description

If the job's pre-execution command fails all attempts, the job is returned to the submission cluster.

Valid values

$0 < \text{MAX_PREEXEC_RETRY} < \text{INFINIT_INT}$

INFINIT_INT is defined in `lsf.h`.

Default

5

MAX_PROTOCOL_INSTANCES

For LSF IBM Parallel Environment (PE) integration. Specifies the number of parallel communication paths (windows) available to the protocol on each network.

Syntax

`MAX_PROTOCOL_INSTANCES=integer`

Description

If number of windows specified for the job (with the `instances` option of `bsub -network` or the `NETWORK_REQ` parameter in `lsb.queues` or `lsb.applications`), or it is greater than the specified maximum value, LSF rejects the job.

Specify `MAX_PROTOCOL_INSTANCES` in a queue (`lsb.queues`) or cluster-wide in `lsb.params`. The value specified in a queue overrides the value specified in `lsb.params`.

`LSF_PE_NETWORK_NUM` must be defined to a non-zero value in `lsf.conf` for `MAX_PROTOCOL_INSTANCES` to take effect and for LSF to run PE jobs. If `LSF_PE_NETWORK_NUM` is not defined or is set to 0, the value of `MAX_PROTOCOL_INSTANCES` is ignored with a warning message.

For best performance, set `MAX_PROTOCOL_INSTANCES` so that the communication subsystem uses every available adapter before it reuses any of the adapters.

Default

No default value

MAX_RSCHED_TIME

Determines how long a MultiCluster job stays pending in the execution cluster before returning to the submission cluster. Used only with the MultiCluster job forwarding model.

Syntax

`MAX_RSCHED_TIME=integer | infinit`

Description

The remote timeout limit in seconds is:

```
MAX_RSCHED_TIME * MBD_SLEEP_TIME=timeout
```

Specify `infinit` to disable remote timeout (jobs always get dispatched in the correct FCFS order because MultiCluster jobs never get rescheduled, but MultiCluster jobs can be pending in the receive-jobs queue forever instead of being rescheduled to a better queue).

Note:

apply to the queue in the submission cluster (only). This parameter is ignored by the receiving queue.

Remote timeout limit never affects advance reservation jobs

Jobs that use an advance reservation always behave as if remote timeout is disabled.

Default

20 (20 minutes by default)

MAX_SLOTS_IN_POOL

Queue-based fairshare only. Specifies the maximum number of job slots available in the slot pool the queue belongs to for queue based fairshare.

Syntax

MAX_SLOTS_IN_POOL=*integer*

Description

Defined in the first queue of the slot pool. Definitions in subsequent queues have no effect.

When defined together with other slot limits (**QJOB_LIMIT**, **HJOB_LIMIT** or **UJOB_LIMIT** in `lsb.queues` or queue limits in `lsb.resources`) the lowest limit defined applies.

When **MAX_SLOTS_IN_POOL**, **SLOT_RESERVE**, and **BACKFILL** are defined for the same queue, jobs in the queue cannot backfill using slots reserved by other jobs in the same queue.

Valid values

MAX_SLOTS_IN_POOL can be any number from 0 to INFINIT_INT, where INFINIT_INT is defined in `lsf.h`.

Default

Not defined

MAX_TOTAL_TIME_PREEMPT

Specifies the accumulated preemption time after which a job cannot be preempted again.

Syntax

MAX_TOTAL_TIME_PREEMPT=*minutes*

where *minutes* is wall-clock time, not normalized time.

Description

Setting the parameter of the same name in `lsb.applications` overrides this parameter; setting this parameter overrides the parameter of the same name in `lsb.params`.

Valid values

Any positive integer greater than or equal to one (1)

Default

Unlimited

MEMLIMIT

Specifies the per-process resident size limit for all job processes from this queue.

Syntax

```
MEMLIMIT=[default_limit] maximum_limit
```

Description

Set this parameter to place a per-process hard process resident set size limit, in KB, for all of the processes belonging to a job from this queue (see **getrlimit(2)**).

Sets the maximum amount of physical memory (resident set size, RSS) that may be allocated to a process.

By default, if a default memory limit is specified, jobs submitted to the queue without a job-level memory limit are killed when the default memory limit is reached.

If you specify only one limit, it is the maximum, or hard, memory limit. If you specify two limits, the first one is the default, or soft, memory limit, and the second one is the maximum memory limit.

LSF has two methods of enforcing memory usage:

- OS Memory Limit Enforcement
- LSF Memory Limit Enforcement

OS memory limit enforcement

OS memory limit enforcement is the default MEMLIMIT behavior and does not require further configuration. OS enforcement usually allows the process to eventually run to completion. LSF passes MEMLIMIT to the OS that uses it as a guide for the system scheduler and memory allocator. The system may allocate more memory to a process if there is a surplus. When memory is low, the system takes memory from and lowers the scheduling priority (re-nice) of a process that has exceeded its declared MEMLIMIT. Only available on systems that support RLIMIT_RSS for **setrlimit()**.

Not supported on:

- Sun Solaris 2.x
- Windows

LSF memory limit enforcement

To enable LSF memory limit enforcement, set LSB_MEMLIMIT_ENFORCE in `lsf.conf` to `y`. LSF memory limit enforcement explicitly sends a signal to kill a running process once it has allocated memory past MEMLIMIT.

You can also enable LSF memory limit enforcement by setting LSB_JOB_MEMLIMIT in `lsf.conf` to `y`. The difference between LSB_JOB_MEMLIMIT set to `y` and LSB_MEMLIMIT_ENFORCE set to `y` is that with LSB_JOB_MEMLIMIT, only the per-job memory limit enforced by LSF is enabled. The per-process memory limit enforced by the OS is disabled. With LSB_MEMLIMIT_ENFORCE set to `y`, both the per-job memory limit enforced by LSF and the per-process memory limit enforced by the OS are enabled.

Available for all systems on which LSF collects total memory usage.

Example

The following configuration defines a queue with a memory limit of 5000 KB:

```
Begin Queue
```

```
QUEUE_NAME = default
```

```
DESCRIPTION = Queue with memory limit of 5000 kbytes
```

```
MEMLIMIT    = 5000
```

```
End Queue
```

Default

Unlimited

MIG

Enables automatic job migration and specifies the migration threshold for checkpointable or rerunnable jobs.

Syntax

MIG=*minutes*

Description

LSF automatically migrates jobs that have been in the SSUSP state for more than the specified number of minutes. Specify a value of 0 to migrate jobs immediately upon suspension. The migration threshold applies to all jobs running on the host.

Job-level command line migration threshold overrides threshold configuration in application profile and queue. Application profile configuration overrides queue level configuration.

When a host migration threshold is specified, and is lower than the value for the job, the queue, or the application, the host value is used..

Members of a chunk job can be migrated. Chunk jobs in WAIT state are removed from the job chunk and put into PEND state.

Does not affect MultiCluster jobs that are forwarded to a remote cluster.

Default

Not defined. LSF does not migrate checkpointable or rerunnable jobs automatically.

NETWORK_REQ

For LSF IBM Parallel Environment (PE) integration. Specifies the network resource requirements for a PE job.

Syntax

NETWORK_REQ="*network_res_req*"

network_res_req has the following syntax:

```
[type=sn_all | sn_single] [:protocol=protocol_name[(protocol_number)]
[, protocol_name[(protocol_number)]] [:mode=US | IP] [:usage=dedicated | shared]
[:instance=positive_integer]
```

Description

If any network resource requirement is specified in the job, queue, or application profile, the job is treated as a PE job. PE jobs can only run on hosts where IBM PE **pnsd** daemon is running.

The network resource requirement string *network_res_req* has the same syntax as the **bsub -network** option.

The `-network bsub` option overrides the value of `NETWORK_REQ` defined in `lsb.queues` or `lsb.applications`. The value of `NETWORK_REQ` defined in `lsb.applications` overrides queue-level `NETWORK_REQ` defined in `lsb.queues`.

The following IBM LoadLeveler job command file options are not supported in LSF:

- `collective_groups`
- `imm_send_buffers`
- `rcxtblocks`

The following network resource requirement options are supported:

type=sn_all | sn_single

Specifies the adapter device type to use for message passing: either `sn_all` or `sn_single`.

sn_single

When used for switch adapters, specifies that all windows are on a single network

sn_all

Specifies that one or more windows are on each network, and that striped communication should be used over all available switch networks. The networks specified must be accessible by all hosts selected to run the PE job. See the *Parallel Environment Runtime Edition for AIX: Operation and Use guide* (SC23-6781-05) for more information about submitting jobs that use striping.

If mode is IP and type is specified as `sn_all` or `sn_single`, the job will only run on InfiniBand (IB) adapters (IPoIB). If mode is IP and type is not specified, the job will only run on Ethernet adapters (IPoEth). For IPoEth jobs, LSF ensures the job is running on hosts where **pnsd** is installed and running. For IPoIB jobs, LSF ensures the job is running on hosts where **pnsd** is installed and running, and that IB networks are up. Because IP jobs do not consume network windows, LSF does not check if all network windows are used up or the network is already occupied by a dedicated PE job.

Equivalent to the PE `MP_EUIDEVICE` environment variable and `-euidvice` PE flag See the *Parallel Environment Runtime Edition for AIX: Operation and Use guide* (SC23-6781-05) for more information. Only `sn_all` or `sn_single` are supported by LSF. The other types supported by PE are not supported for LSF jobs.

protocol=protocol_name[(protocol_number)]

Network communication protocol for the PE job, indicating which message passing API is being used by the application. The following protocols are supported by LSF:

mpi

The application makes only MPI calls. This value applies to any MPI job regardless of the library that it was compiled with (PE MPI, MPICH2).

pami

The application makes only PAMI calls.

lapi

The application makes only LAPI calls.

shmem

The application makes only OpenSHMEM calls.

user_defined_parallel_api

The application makes only calls from a parallel API that you define. For example:
`protocol=myAPI` or `protocol=charm`.

The default value is `mpi`.

LSF also supports an optional *protocol_number* (for example, `mpi(2)`), which specifies the number of contexts (endpoints) per parallel API instance. The number must be a power of 2, but no greater than

128 (1, 2, 4, 8, 16, 32, 64, 128). LSF will pass the communication protocols to PE without any change. LSF will reserve network windows for each protocol.

When you specify multiple parallel API protocols, you cannot make calls to both LAPI and PAMI (`lapi`, `pami`) or LAPI and OpenSHMEM (`lapi`, `shmem`) in the same application. Protocols can be specified in any order.

See the `MP_MSG_API` and `MP_ENDPOINTS` environment variables and the `-msg_api` and `-endpoints` PE flags in the *Parallel Environment Runtime Edition for AIX: Operation and Use* guide (SC23-6781-05) for more information about the communication protocols that are supported by IBM Parallel Edition.

mode=US | IP

The network communication system mode used by the communication specified communication protocol: US (User Space) or IP (Internet Protocol). A US job can only run with adapters that support user space communications, such as the IB adapter. IP jobs can run with either Ethernet adapters or IB adapters. When IP mode is specified, the instance number cannot be specified, and network usage must be unspecified or shared.

Each instance on the US mode requested by a task running on switch adapters requires an adapter window. For example, if a task requests both the MPI and LAPI protocols such that both protocol instances require US mode, two adapter windows will be used.

The default value is US.

usage=dedicated | shared

Specifies whether the adapter can be shared with tasks of other job steps: `dedicated` or `shared`. Multiple tasks of the same job can share one network even if usage is `dedicated`.

The default usage is `shared`.

instances=positive_integer

The number of parallel communication paths (windows) per task made available to the protocol on each network. The number actually used depends on the implementation of the protocol subsystem.

The default value is 1.

If the specified value is greater than `MAX_PROTOCOL_INSTANCES` in `lsb.params` or `lsb.queues`, LSF rejects the job.

`LSF_PE_NETWORK_NUM` must be defined to a non-zero value in `lsf.conf` for `NETWORK_REQ` to take effect. If `LSF_PE_NETWORK_NUM` is not defined or is set to 0, `NETWORK_REQ` is ignored with a warning message.

Example

The following network resource requirement string specifies that the requirements for an `sn_all` job (one or more windows are on each network, and striped communication should be used over all available switch networks). The PE job uses MPI API calls (protocol), runs in user-space network communication system mode, and requires 1 parallel communication path (window) per task.

```
NETWORK_REQ = "protocol=mpi:mode=us:instance=1:type=sn_all"
```

Default

No default value, but if you specify no value (`NETWORK_REQ=""`), the job uses the following: `protocol=mpi:mode=US:usage=shared:instance=1` in the queue.

NEW_JOB_SCHED_DELAY

Specifies the number of seconds that a new job waits, before being scheduled.

Syntax

NEW_JOB_SCHED_DELAY=*seconds*

Description

A value of zero (0) means the job is scheduled without any delay. The scheduler still periodically fetches jobs from mbatchd. Once it gets jobs, scheduler schedules them without any delay. This may speed up job scheduling a bit, but it also generates some communication overhead. Therefore, you should only set it to 0 for high priority, urgent or interactive queues for a small workloads.

If **NEW_JOB_SCHED_DELAY** is set to a non-zero value, scheduler will periodically fetch new jobs from mbatchd, after which it sets job scheduling time to job submission time + **NEW_JOB_SCHED_DELAY**.

Default

0 seconds

NICE

Adjusts the UNIX scheduling priority at which jobs from this queue execute.

Syntax

NICE=*integer*

Description

The default value of 0 (zero) maintains the default scheduling priority for UNIX interactive jobs. This value adjusts the run-time priorities for batch jobs on a queue-by-queue basis, to control their effect on other batch or interactive jobs. See the `nice(1)` manual page for more details.

On Windows, this value is mapped to Windows process priority classes as follows:

- `nice` ≥ 0 corresponds to an priority class of IDLE
- `nice` < 0 corresponds to an priority class of NORMAL

LSF on Windows does not support HIGH or REAL - TIME priority classes.

This value is overwritten by the **NICE** setting in `lsb.applications`, if defined.

Default

0 (zero)

NO_PREEMPT_INTERVAL

Specifies the number of minutes a preemptable job can run before it is preempted. If the uninterrupted run time of a preemptable job is longer than the specified time, it can be preempted.

Syntax

NO_PREEMPT_INTERVAL=*minutes*

The value of *minutes* is wall-clock time, not normalized time.

Description

The `NO_PREEMPT_INTERVAL=0` parameter allows immediate preemption of jobs as soon as they start or resume running.

For example, if a job A needs to preempt other candidate preemptable jobs B, C, and D, the **NO_PREEMPT_INTERVAL** parameter determines which job is preempted:

- Run time of job B and job C is less than the **NO_PREEMPT_INTERVAL** parameter: job B and C are not preempted.
- Run time of job D is greater than or equal to the **NO_PREEMPT_INTERVAL** parameter: job D is preempted.

The parameter of the same name in the `lsb.applications` file overrides this parameter. This parameter overrides the parameter of the same name in the `lsb.params` file.

Default

0

PEND_TIME_LIMIT

Specifies the pending time limit for a job.

Syntax

`PEND_TIME_LIMIT=[hour:]minute`

Description

LSF sends the queue-level pending time limit configuration to IBM Spectrum LSF RTM (LSF RTM), which handles the alarm and triggered actions such as user notification (for example, notifying the user that submitted the job and the LSF administrator) and job control actions (for example, killing the job). LSF RTM compares the job's pending time to the pending time limit, and if the job is pending for longer than this specified time limit, LSF RTM triggers the alarm and actions. This parameter works without LSF RTM, but LSF does not take any other alarm actions.

In MultiCluster job forwarding mode, the job's pending time limit is ignored in the execution cluster, while the submission cluster merges the job's queue-, application-, and job-level pending time limit according to local settings.

The pending time limit is in the form of `[hour:]minute`. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as `3:30`, or `210`.

The job-level pending time limit (**`bsub -pt1`**) overrides the application-level limit (**`PEND_TIME_LIMIT`** in `lsb.applications`), and the application-level limit overrides the queue-level limit specified here.

Default

Not defined.

PJOB_LIMIT

Specifies the per-processor job slot limit for the queue.

Syntax

`PJOB_LIMIT=float`

Description

Maximum number of job slots that this queue can use on any processor. This limit is configured per processor, so that multiprocessor hosts automatically run more jobs.

Default

Unlimited

PLAN

For use when the **ALLOCATION_PLANNER** parameter is enabled. Used to identify the jobs that are candidates for planning.

Syntax

PLAN = Y | N | "<key>[value] ..."

Description

LSF requires that the **ALLOCATION_PLANNER** parameter is enabled in order to use **PLAN=Y**.

Also defined at the cluster and application levels. The precedence is: application, queue, global. For example, application level setting overrides the queue level setting.

The following key-value pairs are supported:

key	value	Default	Description
DELAY	positive integer	-	Number of minutes to delay before considering making a plan for a job following the job's submission time.
MAX_JOBS	positive integer	-	Maximum number of jobs that can have a plan concurrently.

Note:

The **PLAN** parameter replaces the existing **SLOT_RESERVE** parameter and **RESOURCE_RESERVE** parameter when the **ALLOCATION_PLANNER** parameter is enabled.

Default

N

POST_EXEC

Enables post-execution processing at the queue level.

Syntax

POST_EXEC=*command*

Description

The **POST_EXEC** command runs on the execution host after the job finishes. Post-execution commands can be configured at the application and queue levels. Application-level post-execution commands run *before* queue-level post-execution commands.

The **POST_EXEC** command uses the same environment variable values as the job, and, by default, runs under the user account of the user who submits the job. To run post-execution commands under a different user account (such as root for privileged operations), configure the parameter **LSB_PRE_POST_EXEC_USER** in `lsf.sudoers`.

When a job exits with one of the queue's **REQUEUE_EXIT_VALUES**, LSF requeues the job and sets the environment variable **LSB_JOBPEND**. The post-execution command runs after the requeued job finishes.

When the post-execution command is run, the environment variable **LSB_JOBEXIT_STAT** is set to the exit status of the job. If the execution environment for the job cannot be set up, **LSB_JOBEXIT_STAT** is set to 0 (zero).

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

For UNIX:

- The pre- and post-execution commands run in the /tmp directory under /bin/sh -c, which allows the use of shell features in the commands. The following example shows valid configuration lines:

```
PRE_EXEC= /usr/share/lsf/misc/testq_pre >> /tmp/pre.out
```

```
POST_EXEC= /usr/share/lsf/misc/testq_post | grep -v "Hey!"
```

- LSF sets the **PATH** environment variable to

```
PATH='/bin /usr/bin /sbin /usr/sbin'
```

- The stdin, stdout, and stderr are set to /dev/null
- To allow UNIX users to define their own post-execution commands, an LSF administrator specifies the environment variable **\$USER_POSTEXEC** as the **POST_EXEC** command. A user then defines the post-execution command:

```
setenv USER_POSTEXEC /path_name
```

Note: The path name for the post-execution command must be an absolute path. Do not define **POST_EXEC=\$USER_POSTEXEC** when **LSB_PRE_POST_EXEC_USER=root**. This parameter cannot be used to configure host-based post-execution processing.

For Windows:

- The pre- and post-execution commands run under cmd.exe /c
- The standard input, standard output, and standard error are set to NULL
- The **PATH** is determined by the setup of the LSF service

Note:

For post-execution commands that execute on a Windows Server 2003, x64 Edition platform, users must have read and execute privileges for cmd.exe.

Default

Not defined. No post-execution commands are associated with the queue.

PRE_EXEC

Enables pre-execution processing at the queue level.

Syntax

```
PRE_EXEC=command
```

Description

The **PRE_EXEC** command runs on the execution host before the job starts. If the **PRE_EXEC** command exits with a non-zero exit code, LSF requeues the job to the front of the queue.

Pre-execution commands can be configured at the queue, application, and job levels and run in the following order:

1. The queue-level command

- The application-level or job-level command. If you specify a command at both the application and job levels, the job-level command overrides the application-level command; the application-level command is ignored.

The **PRE_EXEC** command uses the same environment variable values as the job, and runs under the user account of the user who submits the job. To run pre-execution commands under a different user account (such as root for privileged operations), configure the parameter **LSB_PRE_POST_EXEC_USER** in `lsf.sudoers`.

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

For UNIX:

- The pre- and post-execution commands run in the `/tmp` directory under `/bin/sh -c`, which allows the use of shell features in the commands. The following example shows valid configuration lines:

```
PRE_EXEC= /usr/share/lsf/misc/testq_pre >> /tmp/pre.out
```

```
POST_EXEC= /usr/share/lsf/misc/testq_post | grep -v "Hey!"
```

- LSF sets the **PATH** environment variable to

```
PATH='/bin /usr/bin /sbin /usr/sbin'
```

- The stdin, stdout, and stderr are set to `/dev/null`

For Windows:

- The pre- and post-execution commands run under `cmd.exe /c`
- The standard input, standard output, and standard error are set to NULL
- The **PATH** is determined by the setup of the LSF Service

Note:

For pre-execution commands that execute on a Windows Server 2003, x64 Edition platform, users must have read and execute privileges for `cmd.exe`. This parameter cannot be used to configure host-based pre-execution processing.

Default

Not defined. No pre-execution commands are associated with the queue.

PREEMPTION

Enables preemptive scheduling and defines this queue as preemptive or preemptable.

Syntax

```
PREEMPTION=PREEMPTIVE[[low_queue_name[+pref_level]...]
```

```
PREEMPTION=PREEMPTABLE[[hi_queue_name...]]
```

```
PREEMPTION=PREEMPTIVE[[low_queue_name[+pref_level]...] PREEMPTABLE[[hi_queue_name...]]
```

Description

PREEMPTIVE

Enables preemptive scheduling and defines this queue as preemptive. Jobs in this queue preempt jobs from the specified lower-priority queues or from all lower-priority queues if the parameter is specified with no queue names. **PREEMPTIVE** can be combined with **PREEMPTABLE** to specify that jobs in this queue can preempt jobs in lower-priority queues, and can be preempted by jobs in higher-priority queues.

PREEMPTABLE

Enables preemptive scheduling and defines this queue as preemptable. Jobs in this queue can be preempted by jobs from specified higher-priority queues, or from all higher-priority queues, even if the higher-priority queues are not preemptive. **PREEMPTIVE** can be combined with **PREEMPTIVE** to specify that jobs in this queue can be preempted by jobs in higher-priority queues, and can preempt jobs in lower-priority queues.

low_queue_name

Specifies the names of lower-priority queues that can be preempted.

To specify multiple queues, separate the queue names with a space, and enclose the list in a single set of square brackets.

+pref_level

Specifies to preempt this queue before preempting other queues. When multiple queues are indicated with a preference level, an order of preference is indicated: queues with higher relative preference levels are preempted before queues with lower relative preference levels set.

hi_queue_name

Specifies the names of higher-priority queues that can preempt jobs in this queue.

To specify multiple queues, separate the queue names with a space and enclose the list in a single set of square brackets.

Example: configure selective, ordered preemption across queues

The following example defines four queues, as follows:

- high
 - Has the highest relative priority of 99
 - Jobs from this queue can preempt jobs from all other queues
- medium
 - Has the second-highest relative priority at 10
 - Jobs from this queue can preempt jobs from `normal` and `low` queues, beginning with jobs from `low`, as indicated by the preference (+1)
- normal
 - Has the second-lowest relative priority, at 5
 - Jobs from this queue can preempt jobs from `low`, and can be preempted by jobs from both `high` and `medium` queues
- low
 - Has the lowest relative priority, which is also the default priority, at 1
 - Jobs from this queue can be preempted by jobs from all preemptive queues, even though it does not have the **PREEMPTABLE** keyword set

```
Begin Queue
QUEUE_NAME=high
PREEMPTION=PREEMPTIVE
PRIORITY=99
End Queue
```

```
Begin Queue
QUEUE_NAME=medium
PREEMPTION=PREEMPTIVE[normal low+1]
PRIORITY=10
End Queue
```

```
Begin Queue
QUEUE_NAME=normal
PREEMPTION=PREEMPTIVE[low]
PREEMPTABLE[high medium]
PRIORITY=5
End Queue
```

```
Begin Queue
QUEUE_NAME=low
PRIORITY=1
End Queue
```

PREEMPT_DELAY

Preemptive jobs will wait the specified number of seconds from the submission time before preempting any low priority preemptable jobs.

Syntax

PREEMPT_DELAY=*seconds*

Description

During the grace period, preemption will not be triggered, but the job can be scheduled and dispatched by other scheduling policies.

This feature can provide flexibility to tune the system to reduce the number of preemptions. It is useful to get better performance and job throughput. When the low priority jobs are short, if high priority jobs can wait a while for the low priority jobs to finish, preemption can be avoided and cluster performance is improved. If the job is still pending after the grace period has expired, the preemption will be triggered.

The waiting time is for preemptive jobs in the pending status only. It will not impact the preemptive jobs that are suspended.

The time is counted from the submission time of the jobs. The submission time means the time **mbatchd** accepts a job, which includes newly submitted jobs, restarted jobs (by **brestart**) or forwarded jobs from a remote cluster.

When the preemptive job is waiting, the pending reason is:

The preemptive job is allowing a grace period before preemption.

If you use an older version of **bjobs**, the pending reason is:

Unknown pending reason code <6701>;

The parameter is defined in `lsb.params`, `lsb.queues` (overrides `lsb.params`), and `lsb.applications` (overrides both `lsb.params` and `lsb.queues`).

Run **badmin reconfig** to make your changes take effect.

Default

Not defined (if the parameter is not defined anywhere, preemption is immediate).

PRIORITY

Specifies the relative queue priority for dispatching jobs. A higher value indicates a higher job-dispatching priority, relative to other queues.

Syntax

PRIORITY=*integer*

Description

LSF schedules jobs from one queue at a time, starting with the highest-priority queue. If multiple queues have the same priority, LSF schedules all the jobs from these queues in first-come, first-served order.

LSF queue priority is independent of the UNIX scheduler priority system for time-sharing processes. In LSF, the NICE parameter is used to set the UNIX time-sharing priority for batch jobs.

integer

Specify a number greater than or equal to 1, where 1 is the lowest priority.

Default

1

PROCESSLIMIT

Limits the number of concurrent processes that can be part of a job.

Syntax

PROCESSLIMIT=[*default_limit*] *maximum_limit*

Description

By default, if a default process limit is specified, jobs submitted to the queue without a job-level process limit are killed when the default process limit is reached.

If you specify only one limit, it is the maximum, or hard, process limit. If you specify two limits, the first one is the default, or soft, process limit, and the second one is the maximum process limit.

Default

Unlimited

QJOB_LIMIT

Specifies the job slot limit for the queue.

Syntax

QJOB_LIMIT=*integer*

Description

This parameter specifies the total number of job slots that this queue can use.

Default

Unlimited

QUEUE_GROUP

Configures absolute priority scheduling (APS) across multiple queues.

Syntax

QUEUE_GROUP=*queue1, queue2 ...*

Description

When APS is enabled in the queue with APS_PRIORITY, the FAIRSHARE_QUEUES parameter is ignored. The QUEUE_GROUP parameter replaces FAIRSHARE_QUEUES, which is obsolete in LSF 7.0.

lsb.queues

Default

Not defined

QUEUE_NAME

Required. Specifies the name of the queue.

Syntax

QUEUE_NAME=*string*

Description

Specify any ASCII string up to 59 characters long. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces. You cannot specify the reserved name `default`.

Default

You must specify this parameter to define a queue. The default queue automatically created by LSF is named `default`.

RC_ACCOUNT

Assigns an account name (tag) to hosts borrowed through LSF resource connector, so that they cannot be used by other user groups, users, or jobs.

Syntax

RC_ACCOUNT=*account_name*

Description

When a job is submitted to a queue with the **RC_ACCOUNT** parameter specified, hosts borrowed to run the job are tagged with the value of the **RC_ACCOUNT** parameter. The borrowed host cannot be used by other queues that have a different value for the **RC_ACCOUNT** parameter (or that don't have the **RC_ACCOUNT** parameter defined).

After the borrowed host joins the cluster, use the **lshosts -s** command to view the value of the **RC_ACCOUNT** parameter for the host.

Example

```
RC_ACCOUNT=project1
```

Default

The string "default" - Meaning, no account is defined for the queue.

RC_DEMAND_POLICY

Defines threshold conditions for the determination of whether demand is triggered to borrow resources through resource connector for all the jobs in a queue. As long as pending jobs at the queue meet at least one threshold condition, LSF expresses the demand to resource connector to trigger borrowing.

Syntax

RC_DEMAND_POLICY = THRESHOLD[[*num_pend_jobs*[,*duration*] ...]

Description

The demand policy defined by the **RC_DEMAND_POLICY** parameter can contain multiple conditions, in an OR relationship. A condition is defined as [*num_pend_jobs*[*duration*]]. The queue has more than the specified number of eligible pending jobs that are expected to run at least the specified duration in minutes. The *num_pend_jobs* option is required, and the duration is optional. The default duration is 0 minutes.

LSF considers eligible pending jobs for the policy. An ineligible pending job (for example, a job dependency is not satisfied yet) keeps pending even though hosts are available. The policy counts a job for eligibility no matter how many tasks or slots the job requires. Each job element is counted as a job. Pending demand for a resizable job is not counted, though LSF can allocate borrowed resources to the resizable job.

LSF evaluates the policies at each demand calculation cycle, and accumulates duration if the *num_pend_jobs* option is satisfied. The **mbschd** daemon resets the duration of the condition when it restarts or if the condition has not been evaluated in the past 2 minutes. For example, if no pending jobs are in the cluster, for 2 minutes, **mbschd** stops evaluating them.

Example

In the following example, LSF calculates demand if the queue has 5 or more pending jobs in past 10 minutes, or 1 or more pending jobs in past 60 minutes, or 100 or more pending jobs.

```
RC_DEMAND_POLICY = THRESHOLD[ [ 5, 10] [1, 60] [100] ]
```

Default

Not defined for the queue

RC_HOSTS

Enables LSF resource connector to borrow specific host types from a resource provider.

Syntax

```
RC_HOSTS=string
```

```
RC_HOSTS = none | all | host_type [host_type ...]
```

Description

The *host_type* flag is a Boolean resource that is a member of the list of host resources that are defined in the **LSB_RC_EXTERNAL_HOST_FLAG** parameter in the `lsf.conf` file.

If the **RC_HOSTS** parameter is not defined in the queue, its default value is none. Borrowing is disabled for any queue that explicitly defines **RC_HOSTS=none**, even if the **LSB_RC_EXTERNAL_HOST_FLAG** parameter is defined in the `lsf.conf` file.

If the **RC_HOSTS** parameter is not defined in any queue, borrowing cannot happen for any job.

Note: The **HOSTS** parameter in the `lsb.queues` file and the **bsub -m** option do not apply to hosts that are managed through the resource connector. To specify the resource connector host types that can be used by a queue, you must specify the **RC_HOSTS** parameter in that queue.

Example

```
RC_HOSTS=awshost
```

Default

none - host borrowing from resource providers is disabled, and no borrowed hosts can be used by the queue.

RCVJOBS_FROM

Defines a receive-jobs queue for LSF multicluster capability.

Syntax

`RCVJOBS_FROM=cluster_name ... | allclusters`

Description

Specify cluster names, separated by a space. The administrator of each remote cluster determines which queues in that cluster forward jobs to the local cluster.

If you enabled an LSF data manager data transfer queue as a remote send-jobs queue in the execution cluster (that is, if you added a queue from the submission cluster to the **SNDJOBS_TO** parameter in the `lsb.queues` file in the execution cluster), you must include the execution cluster in the **RCVJOBS_FROM** parameter in the corresponding submission cluster.

Use the keyword `allclusters` to specify any remote cluster.

Example

The following queue accepts remote jobs from clusters 2, 4, and 6.

```
RCVJOBS_FROM=cluster2 cluster4 cluster6
```

Example for LSF data manager

If the execution cluster `clusterE` has the LSF data manager transfer queue `data_transfer` set as a remote send-jobs queue to the `receive_q` queue in the submission cluster `clusterS`, according to the following configuration for the `clusterE` cluster:

```
Begin Queue
QUEUE_NAME=data_transfer
DATA_TRANSFER=Y
SNDJOBS_TO=receive_q@clusterS
HOSTS=hostS1 hostS2 # Transfer nodes in the execution cluster
End Queue
```

You must define the **RCVJOBS_FROM** parameter for the `receive_q` queue in the submission cluster `clusterS` to accept jobs from (and push data to) the execution cluster `clusterE`, as shown in the following configuration for the `clusterS` cluster:

```
Begin Queue
QUEUE_NAME=receive_q
RCVJOBS_FROM=clusterE
PRIORITY=40
HOSTS=hostS1 hostS2 # Transfer nodes in the submission cluster
RES_REQ=select[type==any]
End Queue
```

Alternatively, you can define `RCVJOBS_FROM=allclusters` to accept jobs from all clusters, which includes the execution cluster.

RELAX_JOB_DISPATCH_ORDER

Allows LSF to deviate from standard job prioritization policies to improve cluster utilization.

Syntax

```
RELAX_JOB_DISPATCH_ORDER=Y | y | N | n | ALLOC_REUSE_DURATION[[min] max]
```

Description

When this parameter is enabled, LSF allows multiple jobs with common resource requirements to run consecutively on the same allocation. Whenever a job finishes, LSF attempts to quickly replace it with a pending job that has the same resource requirements. To ensure that limits are not violated, LSF selects pending jobs that belong to the same user and have other attributes in common.

Since LSF bypasses most of the standard scheduling logic between jobs, reusing resource allocation can help improve cluster utilization. This improvement is most evident in clusters with several shorter jobs (that is, jobs that run from a few seconds to several minutes).

To ensure that the standard job prioritization policies are approximated, there is a limit on the length of time that each allocation is reusable. LSF automatically sets this time limit to achieve a high level of resource utilization. By default, this reuse time cannot exceed 30 minutes. If you specify a maximum reuse time and an optional minimum reuse time (by using **ALLOC_REUSE_DURATION**), LSF adjusts the time limit within this specified range to achieve the highest level of resource utilization.

If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Examples

- `RELAX_JOB_DISPATCH_ORDER=Y`

The resource allocation of a finished job can be reused from 0 to 30 minutes.

- `RELAX_JOB_DISPATCH_ORDER=ALLOC_REUSE_DURATION[45]`

The resource allocation of a finished job can be reused from 0 to 45 minutes.

- `RELAX_JOB_DISPATCH_ORDER=ALLOC_REUSE_DURATION[5 45]`

The resource allocation of a finished job can be reused from 5 to 45 minutes.

Default

Not defined.

REMOTE_MAX_PREEEXEC_RETRY

Define the maximum number of times to attempt the pre-execution command of a job from the remote cluster. Used only with the MultiCluster job forwarding model.

Syntax

```
REMOTE_MAX_PREEEXEC_RETRY=integer
```

Description

This parameter applies to the execution cluster.

Valid values

0 - INFINIT_INT

INFINIT_INT is defined in `lsf.h`.

Default

5

REQUEUE_EXIT_VALUES

Enables automatic job requeue and sets the LSB_EXIT_REQUEUE environment variable.

Syntax

```
REQUEUE_EXIT_VALUES=[exit_code ...] [EXCLUDE(exit_code ...)]
```

Description

Use spaces to separate multiple exit codes. Application-level exit values override queue-level values. Job-level exit values (**bsub -Q**) override application-level and queue-level values.

exit_code has the following form:

```
"[all] [~number ...] | [number ...]"
```

The reserved keyword all specifies all exit codes. Exit codes are typically between 0 and 255. Use a tilde (~) to exclude specified exit codes from the list.

Jobs are requeued to the head of the queue. The output from the failed run is not saved, and the user is not notified by LSF.

Define an exit code as EXCLUDE(*exit_code*) to enable exclusive job requeue, ensuring the job does not rerun on the samehost. Exclusive job requeue does not work for parallel jobs.

For MultiCluster jobs forwarded to a remote execution cluster, the exit values specified in the submission cluster with the EXCLUDE keyword are treated as if they were non-exclusive.

You can also requeue a job if the job is terminated by a signal.

If a job is killed by a signal, the exit value is $128 + \text{signal_value}$. The sum of 128 and the signal value can be used as the exit code in the parameter REQUEUE_EXIT_VALUES.

For example, if you want a job to rerun if it is killed with a signal 9 (SIGKILL), the exit value would be $128 + 9 = 137$. You can configure the following requeue exit value to allow a job to be requeue if it was kill by signal 9:

```
REQUEUE_EXIT_VALUES=137
```

In Windows, if a job is killed by a signal, the exit value is *signal_value*. The signal value can be used as the exit code in the parameter REQUEUE_EXIT_VALUES.

For example, if you want to rerun a job after it was killed with a signal 7 (SIGKILL), the exit value would be 7. You can configure the following requeue exit value to allow a job to requeue after it was killed by signal 7:

```
REQUEUE_EXIT_VALUES=7
```

You can configure the following requeue exit value to allow a job to requeue for both Linux and Windows after it was killed:

```
REQUEUE_EXIT_VALUES=137 7
```

If mbatchd is restarted, it does not remember the previous hosts from which the job exited with an exclusive requeue exit code. In this situation, it is possible for a job to be dispatched to hosts on which the job has previously exited with an exclusive exit code.

You should configure REQUEUE_EXIT_VALUES for interruptible backfill queues (INTERRUPTIBLE_BACKFILL=*seconds*).

Example

```
REQUEUE_EXIT_VALUES=30 EXCLUDE(20)
```

means that jobs with exit code 30 are requeued, jobs with exit code 20 are requeued exclusively, and jobs with any other exit code are not requeued.

Default

Not defined. Jobs are not requeued.

RERUNNABLE

Enables automatic rerun for jobs from this queue.

Syntax

```
RERUNNABLE=yes | no
```

Description

If set to yes, enables automatic job rerun (restart).

Rerun is disabled when RERUNNABLE is set to no. The yes and no arguments are not case sensitive.

For MultiCluster jobs, the setting in the submission queue is used, and the setting in the execution queue is ignored.

Members of a chunk job can be rerunnable. If the execution host becomes unavailable, rerunnable chunk job members are removed from the job chunk and dispatched to a different execution host.

Default

no

RESOURCE_RESERVE

Enables processor reservation and memory reservation for pending jobs for this queue.

Syntax

```
RESOURCE_RESERVE=MAX_RESERVE_TIME [integer]
```

Description

Specifies the number of dispatch turns (MAX_RESERVE_TIME) over which a job can reserve job slots and memory.

Overrides the SLOT_RESERVE parameter. If both RESOURCE_RESERVE and SLOT_RESERVE are defined in the same queue, an error is displayed when the cluster is reconfigured, and SLOT_RESERVE is ignored. Job slot reservation for parallel jobs is enabled by RESOURCE_RESERVE if the LSF scheduler plug-in module names for both resource reservation and parallel batch jobs (schmod_parallel and schmod_reserve) are configured in the lsb.modules file: The schmod_parallel name *must* come before schmod_reserve in lsb.modules.

If a job has not accumulated enough memory or job slots to start by the time MAX_RESERVE_TIME expires, it releases all its reserved job slots or memory so that other pending jobs can run. After the reservation time expires, the job cannot reserve memory or slots for one scheduling session, so other jobs have a chance to be dispatched. After one scheduling session, the job can reserve available memory and job slots again for another period specified by MAX_RESERVE_TIME.

If BACKFILL is configured in a queue, and a run limit is specified with -W on **bsub** or with RUNLIMIT in the queue, backfill jobs can use the accumulated memory reserved by the other jobs in the queue, as long as the backfill job can finish before the predicted start time of the jobs with the reservation.

Unlike slot reservation, which only applies to parallel jobs, memory reservation and backfill on memory apply to sequential and parallel jobs.

Example

```
RESOURCE_RESERVE=MAX_RESERVE_TIME[5]
```

This example specifies that jobs have up to 5 dispatch turns to reserve sufficient job slots or memory (equal to 5 minutes, by default).

Default

Not defined. No job slots or memory is reserved.

RES_REQ

Specifies resource requirements used to determine eligible hosts.

Syntax

```
RES_REQ=res_req
```

Description

Specify a resource requirement string as usual. The resource requirement string lets you specify conditions in a more flexible manner than using the load thresholds. Resource requirement strings can be simple (applying to the entire job), compound (applying to the specified number of slots) or can contain alternative resources (alternatives between 2 or more simple and/or compound). For alternative resources, if the first resource cannot be found that satisfies the first resource requirement, then the next resource requirement is tried, and so on until the requirement is satisfied.

Compound and alternative resource requirements follow the same set of rules for determining how resource requirements are going to be merged between job, application, and queue level. For more information on merge rules, see *Administering IBM Spectrum LSF*.

When a compound or alternative resource requirement is set for a queue, it will be ignored unless it is the only resource requirement specified (no resource requirements are set at the job-level or application-level).

When a simple resource requirement is set for a queue and a compound resource requirement is set at the job-level or application-level, the queue-level requirements merge as they do for simple resource requirements. However, any job-based resources defined in the queue only apply to the first term of the merged compound resource requirements.

When `LSF_STRICT_RESREQ=Y` is configured in `lsf.conf`, resource requirement strings in `select` sections must conform to a more strict syntax. The strict resource requirement syntax only applies to the `select` section. It does not apply to the other resource requirement sections (`order`, `rusage`, `same`, `span`, `cu` or `affinity`). When `LSF_STRICT_RESREQ=Y` in `lsf.conf`, LSF rejects resource requirement strings where an `rusage` section contains a non-consumable resource.

For simple resource requirements, the `select` sections from all levels must be satisfied and the same sections from all levels are combined. `cu`, `order`, and `span` sections at the job-level overwrite those at the application-level which overwrite those at the queue-level. Multiple `rusage` definitions are merged, with the job-level `rusage` taking precedence over the application-level, and application-level taking precedence over the queue-level.

The simple resource requirement `rusage` section can specify additional requests. To do this, use the `OR (|)` operator to separate additional `rusage` strings. Multiple `-R` options cannot be used with multi-phase `rusage` resource requirements.

For simple resource requirements the job-level `affinity` section overrides the application-level, and the application-level `affinity` section overrides the queue-level.

Note:

Compound and alternative resource requirements do not support use of the || operator within rusage sections or the cu section.

The **RES_REQ** consumable resource requirements must satisfy any limits set by the parameter **RESRSV_LIMIT** in `lsb.queues`, or the **RES_REQ** will be ignored.

When both the **RES_REQ** and **RESRSV_LIMIT** are set in `lsb.queues` for a consumable resource, the queue-level **RES_REQ** no longer acts as a hard limit for the merged **RES_REQ** rusage values from the job and application levels. In this case only the limits set by **RESRSV_LIMIT** must be satisfied, and the queue-level **RES_REQ** acts as a default value.

For example:

Queue-level RES_REQ:

```
RES_REQ=rusage[mem=200:lic=1] ...
```

For the job submission:

```
bsub -R'rusage[mem=100]' ...
```

the resulting requirement for the job is

```
rusage[mem=100:lic=1]
```

where `mem=100` specified by the job overrides `mem=200` specified by the queue. However, `lic=1` from queue is kept, since job does not specify it.

Queue-level RES_REQ threshold:

```
RES_REQ = rusage[bwidth =2:threshold=5] ...
```

For the job submission:

```
bsub -R "rusage[bwidth =1:threshold=6]" ...
```

the resulting requirement for the job is

```
rusage[bwidth =1:threshold=6]
```

Queue-level RES_REQ with decay and duration defined:

```
RES_REQ=rusage[mem=200:duration=20:decay=1] ...
```

For a job submission with no decay or duration:

```
bsub -R'rusage[mem=100]' ...
```

the resulting requirement for the job is:

```
rusage[mem=100:duration=20:decay=1]
```

Queue-level duration and decay are merged with the job-level specification, and `mem=100` for the job overrides `mem=200` specified by the queue. However, `duration=20` and `decay=1` from queue are kept, since job does not specify them.

Queue-level RES_REQ with resource reservation method:

```
RES_REQ=rusage[mem=200/host:duration=20:decay=1] ...
```

For a job submission with no decay or duration:

```
bsub -R'rusage[mem=100/task]' ...
```

the resulting requirement for the job is:

```
rusage[mem=100/task:duration=20:decay=1]
```

Queue-level RES_REQ with multi-phase job-level rusage:

```
RES_REQ=rusage[mem=200:duration=20:decay=1] ...
```

For a job submission with no decay or duration:

```
bsub -R'rusage[mem=(300 200 100):duration=(10 10 10)]' ...
```

the resulting requirement for the job is:

```
rusage[mem=(300 200 100):duration=(10 10 10)]
```

Multi-phase rusage values in the job submission override the single phase specified by the queue.

- If **RESRSV_LIMIT** is defined in `lsb.queues` and has a maximum memory limit of 300 MB or greater, this job will be accepted.
- If **RESRSV_LIMIT** is defined in `lsb.queues` and has a maximum memory limit of less than 300 MB, this job will be rejected.
- If **RESRSV_LIMIT** is not defined in `lsb.queues` and the queue-level **RES_REQ** value of 200 MB acts as a ceiling, this job will be rejected.

Queue-level multi-phase rusage RES_REQ:

```
RES_REQ=rusage[mem=(350 200):duration=(20):decay=(1)] ...
```

For a single phase job submission with no decay or duration:

```
bsub -q q_name -R'rusage[mem=100:swap=150]' ...
```

the resulting requirement for the job is:

```
rusage[mem=100:swap=150]
```

The job-level rusage string overrides the queue-level multi-phase rusage string.

The `order` section defined at the job level overwrites any resource requirements specified at the application level or queue level. The `order` section defined at the application level overwrites any resource requirements specified at the queue level. The default `order` string is `r15s:pg`.

If **RES_REQ** is defined at the queue level and there are no load thresholds defined, the pending reasons for each individual load index are not displayed by **bjobs**.

The `span` section defined at the queue level is ignored if the `span` section is also defined at the job level or in an application profile.

Note: Define `span[hosts=-1]` in the application profile or **bsub -R** resource requirement string to override the `span` section setting in the queue.

Default

`select[type==local] order[r15s:pg]`. If this parameter is defined and a host model or Boolean resource is specified, the default type is any.

RESRSV_LIMIT

Sets a range of allowed values for **RES_REQ** resources.

Syntax

```
RESRSV_LIMIT=[res1={min1,} max1] [res2={min2,} max2]...
```

Where *res* is a consumable resource name, *min* is an optional minimum value and *max* is the maximum allowed value. Both *max* and *min* must be float numbers between 0 and 2147483647, and *min* cannot be greater than *max*.

Description

Queue-level **RES_REQ** usage values (set in `lsb.queues`) must be in the range set by **RESRSV_LIMIT**, or the queue-level **RES_REQ** is ignored. Merged **RES_REQ** usage values from the job and application levels must be in the range of **RESRSV_LIMIT**, or the job is rejected.

Changes made to the usage values of running jobs using **bmod -R** cannot exceed the maximum values of **RESRSV_LIMIT**, but can be lower than the minimum values.

When both the **RES_REQ** and **RESRSV_LIMIT** are set in `lsb.queues` for a consumable resource, the queue-level **RES_REQ** no longer acts as a hard limit for the merged **RES_REQ** usage values from the job and application levels. In this case only the limits set by **RESRSV_LIMIT** must be satisfied, and the queue-level **RES_REQ** acts as a default value.

For MultiCluster, jobs must satisfy the **RESRSV_LIMIT** range set for the send-jobs queue in the submission cluster. After the job is forwarded the resource requirements are also checked against the **RESRSV_LIMIT** range set for the receive-jobs queue in the execution cluster.

Note:

Only consumable resource limits can be set in **RESRSV_LIMIT**. Other resources will be ignored.

Default

Not defined.

If *max* is defined and optional *min* is not, the default for *min* is 0.

RESUME_COND

Specifies conditions for LSF to automatically resume a suspended (SSUSP) job in this queue.

Syntax

`RESUME_COND=res_req`

Use the `select` section of the resource requirement string to specify load thresholds. All other sections are ignored.

Description

LSF automatically resumes a suspended (SSUSP) job in this queue if the load on the host satisfies the specified conditions. The conditions only support load indices and static boolean resources.

If **RESUME_COND** is not defined, then the `loadSched` thresholds are used to control resuming of jobs. The `loadSched` thresholds are ignored, when resuming jobs, if **RESUME_COND** is defined.

Default

Not defined. The `loadSched` thresholds are used to control resuming of jobs.

RUN_JOB_FACTOR

Job slots weighting factor. Used only with fairshare scheduling.

Syntax

`RUN_JOB_FACTOR=number`

Description

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the number of job slots reserved and in use by a user.

If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Default

Not defined.

RUN_TIME_DECAY

Enables decay for run time at the same rate as the decay set by `HIST_HOURS` for cumulative CPU time and historical run time. Used only with fairshare scheduling.

Syntax

`RUN_TIME_DECAY=Y | y | N | n`

Description

In the calculation of a user's dynamic share priority, this factor determines whether run time is decayed.

If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Restrictions

Running `badmin reconfig` or restarting `mbatchd` during a job's run time results in the decayed run time being recalculated.

When a suspended job using run time decay is resumed, the decay time is based on the elapsed time.

Default

Not defined

RUN_TIME_FACTOR

Specifies the run time weighting factor. Used only with fairshare scheduling.

Syntax

`RUN_TIME_FACTOR=number`

Description

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the total run time of a user's running jobs.

If undefined, the cluster-wide value from the `lsb.params` parameter of the same name is used.

Default

Not defined.

RUN_WINDOW

Specifies time periods during which jobs in the queue are allowed to run.

Syntax

`RUN_WINDOW=time_window ...`

Description

When the window closes, LSF suspends jobs running in the queue and stops dispatching jobs from the queue. When the window reopens, LSF resumes the suspended jobs and begins dispatching additional jobs.

Default

Not defined. Queue is always active.

RUNLIMIT

Specifies the maximum run limit and optionally the default run limit. The name of a host or host model specifies the runtime normalization host to use.

Syntax

```
RUNLIMIT=[default_limit] maximum_limit
```

where *default_limit* and *maximum_limit* are:

```
[hour:]minute[/host_name | /host_model]
```

Description

By default, jobs that are in a running state (but not in pre-execution or post-execution) for longer than the specified maximum run limit are killed by LSF. You can optionally provide your own termination job action to override this default.

Jobs submitted with a job-level run limit (**bsub -W**) that is less than the maximum run limit are killed when their job-level run limit is reached. Jobs submitted with a run limit greater than the maximum run limit are rejected by the queue.

If a default run limit is specified, jobs submitted to the queue without a job-level run limit are killed when the default run limit is reached. The default run limit is used with backfill scheduling of parallel jobs.

Note:

If you want to provide an estimated run time for scheduling purposes without killing jobs that exceed the estimate, define the `RUNTIME` parameter in an application profile instead of a run limit (see `lsb.applications` for details).

If you specify only one limit, it is the maximum, or hard, run limit. If you specify two limits, the first one is the default, or soft, run limit, and the second one is the maximum run limit.

The run limit is in the form of `[hour:]minute`. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as `3:30`, or `210`.

The run limit you specify is the normalized run time. This is done so that the job does approximately the same amount of processing, even if it is sent to host with a faster or slower CPU. Whenever a normalized run time is given, the actual time on the execution host is the specified time multiplied by the CPU factor of the normalization host then divided by the CPU factor of the execution host.

If `ABS_RUNLIMIT=Y` is defined in `lsb.params`, the runtime limit is not normalized by the host CPU factor. Absolute wall-clock run time is used for all jobs submitted to a queue with a run limit configured.

Optionally, you can supply a host name or a host model name defined in LSF. You must insert `/` between the run limit and the host name or model name. (See `lsinfo(1)` to get host model information.)

If no host or host model is given, LSF uses the default runtime normalization host defined at the queue level (`DEFAULT_HOST_SPEC` in `lsb.queues`) if it has been configured; otherwise, LSF uses the default CPU time normalization host defined at the cluster level (`DEFAULT_HOST_SPEC` in `lsb.params`) if it has been configured; otherwise, the host with the largest CPU factor (the fastest host in the cluster).

lsb.queues

For MultiCluster jobs, if no other CPU time normalization host is defined and information about the submission host is not available, LSF uses the host with the largest CPU factor (the fastest host in the cluster).

Jobs submitted to a chunk job queue are not chunked if RUNLIMIT is greater than 30 minutes.

RUNLIMIT is required for queues configured with INTERRUPTIBLE_BACKFILL.

Default

Unlimited

SLA_GUARANTEES_IGNORE

Applies to SLA guarantees only. Allows jobs in the queue access to all guaranteed resources.

Syntax

SLA_GUARANTEES_IGNORE=Y | y | N | n

Description

SLA_GUARANTEES_IGNORE=Y allows jobs in the queue access to all guaranteed resources. As a result, some guarantees might not be honored. If a queue does not have this parameter set, jobs in this queue cannot trigger preemption of an SLA job. If an SLA job is suspended (e.g. by a **bstop**), jobs in queues without the parameter set can still make use of the slots released by the suspended job.

Note: Using SLA_GUARANTEES_IGNORE=Y defeats the purpose of guaranteeing resources. This should be used sparingly for low traffic queues only.

Default

Not defined (N). The queue must honor resource guarantees when dispatching jobs.

SLOT_POOL

Specifies the name of the pool of job slots the queue belongs to for queue-based fairshare.

Syntax

SLOT_POOL=*pool_name*

Description

A queue can only belong to one pool. All queues in the pool must share the same set of hosts.

Valid values

Specify any ASCII string up to 60 characters long. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces.

Default

Not defined. No job slots are reserved.

SLOT_RESERVE

Enables processor reservation for the queue and specifies the reservation time.

Syntax

SLOT_RESERVE=MAX_RESERVE_TIME [*integer*]

Description

Specify the keyword `MAX_RESERVE_TIME` and, in square brackets, the number of `MBD_SLEEP_TIME` cycles over which a job can reserve job slots. `MBD_SLEEP_TIME` is defined in `lsb.params`; the default value is 60 seconds.

If a job has not accumulated enough job slots to start before the reservation expires, it releases all its reserved job slots so that other jobs can run. Then, the job cannot reserve slots for one scheduling session, so other jobs have a chance to be dispatched. After one scheduling session, the job can reserve job slots again for another period specified by `SLOT_RESERVE`.

`SLOT_RESERVE` is overridden by the `RESOURCE_RESERVE` parameter.

If both `RESOURCE_RESERVE` and `SLOT_RESERVE` are defined in the same queue, job slot reservation and memory reservation are enabled and an error is displayed when the cluster is reconfigured. `SLOT_RESERVE` is ignored.

Job slot reservation for parallel jobs is enabled by `RESOURCE_RESERVE` if the LSF scheduler plug-in module names for both resource reservation and parallel batch jobs (`schmod_parallel` and `schmod_reserve`) are configured in the `lsb.modules` file: The `schmod_parallel` name *must* come before `schmod_reserve` in `lsb.modules`.

If `BACKFILL` is configured in a queue, and a run limit is specified at the job level (`bsub -W`), application level (`RUNLIMIT` in `lsb.applications`), or queue level (`RUNLIMIT` in `lsb.queues`), or if an estimated run time is specified at the application level (`RUNTIME` in `lsb.applications`), backfill parallel jobs can use job slots reserved by the other jobs, as long as the backfill job can finish before the predicted start time of the jobs with the reservation.

Unlike memory reservation, which applies both to sequential and parallel jobs, slot reservation applies only to parallel jobs.

Example

```
SLOT_RESERVE=MAX_RESERVE_TIME[5]
```

This example specifies that parallel jobs have up to 5 cycles of `MBD_SLEEP_TIME` (5 minutes, by default) to reserve sufficient job slots to start.

Default

Not defined. No job slots are reserved.

SLOT_SHARE

Specifies the share of job slots for queue-based fairshare.

Syntax

```
SLOT_SHARE=integer
```

Description

Represents the percentage of running jobs (job slots) in use from the queue. `SLOT_SHARE` must be greater than zero (0) and less than or equal to 100.

The sum of `SLOT_SHARE` for all queues in the pool does not need to be 100%. It can be more or less, depending on your needs.

Default

Not defined

SNDJOBS_TO

Defines a send-jobs queue for IBM Spectrum LSF multicluster capability.

Syntax

```
SNDJOBS_TO=[queue@]cluster_name[+preference] ...
```

Description

Specify remote queue names, in the form `queue_name@cluster_name[+preference]`, separated by a space.

This parameter is ignored if `lsb.queues HOSTS` specifies remote (borrowed) resources.

Queue preference is defined at the queue level in **SNDJOBS_TO** of the submission cluster for each corresponding execution cluster queue receiving forwarded jobs.

You can enable an LSF data manager data transfer queue as a remote send-jobs queue in the execution cluster.

If you specify a remote queue with the **SNDJOBS_TO** parameter in the data transfer queue in the execution cluster, the path of the **FILE_TRANSFER_CMD** parameter must exist in the submission cluster. In addition, the corresponding remote queue in the submission cluster must be configured to receive the data transfer job (that is, the **RCVJOBS_FROM** parameter value for the remote queue in the submission cluster either includes this execution cluster, or is set to `allclusters`). This ensures that the submission cluster can push data files back to the execution cluster.

Example

```
SNDJOBS_TO=queue2@cluster2+1 queue3@cluster2+2
```

Example for LSF data manager

The following configuration on the execution cluster `clusterE` has the LSF data manager transfer queue `data_transfer` set as a remote send-jobs queue to the `receive_q` queue in the submission cluster `clusterS`:

```
Begin Queue
QUEUE_NAME=data_transfer
DATA_TRANSFER=Y
SNDJOBS_TO=receive_q@clusterS
HOSTS=hostS1 hostS2 # Transfer nodes in the execution cluster
End Queue
```

You must also define the **RCVJOBS_FROM** parameter for the `receive_q` queue in the submission cluster `clusterS` to accept jobs from (and push data to) the execution cluster `clusterE`, as shown in the following configuration in the `clusterS` cluster:

```
Begin Queue
QUEUE_NAME=receive_q
RCVJOBS_FROM=clusterE
PRIORITY=40
HOSTS=hostS1 hostS2 # Transfer nodes in the submission cluster
RES_REQ=select[type==any]
End Queue
```

Alternatively, you can define `RCVJOBS_FROM=allclusters` to accept jobs from all clusters, which includes the execution cluster.

STACKLIMIT

Specifies the per-process stack segment size limit for all job processes from this queue.

Syntax

STACKLIMIT=*integer*

Description

Specify this parameter to place a per-process hard stack segment size limit, in KB, for all of the processes belonging to a job from this queue (see **getrlimit(2)**).

Default

Unlimited

STOP_COND

Specifies conditions for LSF to automatically suspend a running job in this queue.

Syntax

STOP_COND=*res_req*

Use the `select` section of the resource requirement string to specify load thresholds. All other sections are ignored.

Description

LSF automatically suspends a running job in this queue if the load on the host satisfies the specified conditions. The conditions only support load indices and static boolean resources.

- LSF does not suspend the only job running on the host if the machine is interactively idle (`it > 0`).
- LSF does not suspend a forced job (**brun -f**).
- LSF does not suspend a job because of paging rate if the machine is interactively idle.

If STOP_COND is specified in the queue and there are no load thresholds, the suspending reasons for each individual load index is not displayed by **bjobs**.

Example

```
STOP_COND= select[(!cs && it < 5) || (cs && mem < 15 && swp < 50)]
```

In this example, assume “cs” is a Boolean resource indicating that the host is a computer server. The stop condition for jobs running on computer servers is based on the availability of swap memory. The stop condition for jobs running on other kinds of hosts is based on the idle time.

SUCCESS_EXIT_VALUES

Specifies exit values used by LSF to determine if the job was done successfully.

Syntax

SUCCESS_EXIT_VALUES=[*exit_code ...*]

Description

Application level success exit values defined with **SUCCESS_EXIT_VALUES** in `lsb.applications` override the configuration defined in `lsb.queues`. Job-level success exit values specified with the **LSB_SUCCESS_EXIT_VALUES** environment variable override the configuration in `lsb.queues` and `lsb.applications`.

Use **SUCCESS_EXIT_VALUES** for submitting jobs to specific queues that successfully exit with non-zero values so that LSF does not interpret non-zero exit codes as job failure.

If the same exit code is defined in **SUCCESS_EXIT_VALUES** and **REQUEUE_EXIT_VALUES**, any job with this exit code is requeued instead of being marked as DONE because sbatchd processes requeue exit values before success exit values.

In MultiCluster job forwarding mode, LSF uses the **SUCCESS_EXIT_VALUES** from the remote cluster.

In a MultiCluster resource leasing environment, LSF uses the **SUCCESS_EXIT_VALUES** from the consumer cluster.

exit_code should be a value between 0 and 255. Use spaces to separate multiple exit codes.

Any changes you make to **SUCCESS_EXIT_VALUES** will not affect running jobs. Only pending jobs will use the new **SUCCESS_EXIT_VALUES** definitions, even if you run **admin reconfig** and **mbatchd restart** to apply your changes.

Default

Not defined.

SWAPLIMIT

Specifies the amount of total virtual memory limit, in KB, for a job from this queue.

Syntax

SWAPLIMIT=*integer*

Description

This limit applies to the whole job, no matter how many processes the job may contain.

The action taken when a job exceeds its SWAPLIMIT or PROCESSLIMIT is to send SIGQUIT, SIGINT, SIGTERM, and SIGKILL in sequence. For CPULIMIT, SIGXCPU is sent before SIGINT, SIGTERM, and SIGKILL.

Default

Unlimited

TASKLIMIT

Specifies the maximum number of tasks that can be allocated to a job. For parallel jobs, the maximum number of tasks that can be allocated to the job.

Syntax

TASKLIMIT=[*minimum_limit* [*default_limit*]] *maximum_limit*

Description

Note: **TASKLIMIT** replaces **PROCLIMIT** as of LSF 9.1.3.

Queue level **TASKLIMIT** has the highest priority over application level **TASKLIMIT** and job level **TASKLIMIT**. Application level **TASKLIMIT** has higher priority than job level **TASKLIMIT**. Job-level limits must fall within the maximum and minimum limits of the application profile and the queue.

Note: If you also defined **JOB_SIZE_LIST** in the same queue where you defined **TASKLIMIT**, the **TASKLIMIT** parameter is ignored.

Optionally specifies the minimum and default number of job tasks.

All limits must be positive numbers greater than or equal to 1 that satisfy the following relationship:

$1 \leq \text{minimum} \leq \text{default} \leq \text{maximum}$

If **RES_REQ** in a queue was defined as a compound resource requirement with a block size (span[block=value]), the default value for **TASKLIMIT** should be a multiple of a block.

For example, this configuration would be accepted:

```
Queue-level RES_REQ="1*{type==any } + {type==local span[block=4]}"
TASKLIMIT = 5 9 13
```

This configuration, for example, would not be accepted. An error message will appear when doing **badadmin reconfig**:

```
Queue-level RES_REQ="1*{type==any } + {type==local span[block=4]}"
TASKCLIMIT = 4 10 12
```

In the MultiCluster job forwarding model, the local cluster considers the receiving queue's **TASKLIMIT** on remote clusters before forwarding jobs. If the receiving queue's **TASKLIMIT** definition in the remote cluster cannot satisfy the job's task requirements for a remote queue, the job is not forwarded to that remote queue in the cluster.

Default

Unlimited, the default number of tasks is 1

TERMINATE_WHEN

Specifies the circumstances under which the queue invokes the TERMINATE action instead of the SUSPEND action.

Syntax

```
TERMINATE_WHEN=[LOAD] [PREEMPT] [WINDOW]
```

Description

Configures the queue to invoke the TERMINATE action instead of the SUSPEND action in the specified circumstance:

- **LOAD**: kills jobs when the load exceeds the suspending thresholds.
- **PREEMPT**: kills jobs that are being preempted.
- **WINDOW**: kills jobs if the run window closes.

If the TERMINATE_WHEN job control action is applied to a chunk job, sbatchd kills the chunk job element that is running and puts the rest of the waiting elements into pending state to be rescheduled later.

Example

Set TERMINATE_WHEN to WINDOW to define a night queue that kills jobs if the run window closes:

```
Begin Queue
NAME           = night
RUN_WINDOW     = 20:00-08:00 EDT
TERMINATE_WHEN = WINDOW
JOB_CONTROLS   = TERMINATE[kill -KILL $LS_JOBPGIDS; mail - s "job $LSB_JOBID
                killed by queue run window" $USER < /dev/null]
End Queue
```

Specifying the time zone is optional. If you do not specify a time zone, LSF uses the local system time zone. LSF supports the following time zones:

THREADLIMIT

Limits the number of concurrent threads that can be part of a job. Exceeding the limit causes the job to terminate.

Syntax

```
THREADLIMIT=[default_limit] maximum_limit
```

Description

The system sends the following signals in sequence to all processes belongs to the job: SIGINT, SIGTERM, and SIGKILL.

By default, if a default thread limit is specified, jobs submitted to the queue without a job-level thread limit are killed when the default thread limit is reached.

If you specify only one limit, it is the maximum, or hard, thread limit. If you specify two limits, the first one is the default, or soft, thread limit, and the second one is the maximum thread limit.

Both the default and the maximum limits must be positive integers. The default limit must be less than the maximum limit. The default limit is ignored if it is greater than the maximum limit.

Examples

```
THREADLIMIT=6
```

No default thread limit is specified. The value 6 is the default and maximum thread limit.

```
THREADLIMIT=6 8
```

The first value (6) is the default thread limit. The second value (8) is the maximum thread limit.

Default

Unlimited

UJOB_LIMIT

Specifies the per-user job slot limit for the queue.

Syntax

```
UJOB_LIMIT=integer
```

Description

This parameter specifies the maximum number of job slots that each user can use in this queue.

UJOB_LIMIT must be within or greater than the range set by **TASKLIMIT** or **bsub -n** (if either is used), or jobs are rejected.

Default

Unlimited

USE_PAM_CREDS

Applies PAM limits to this queue.

Syntax

```
USE_PAM_CREDS=y | n
```

Description

USE_PAM_CREDS is only supported on Linux systems.

If **USE_PAM_CREDS** is set to **y**, LSF applies PAM limits to a queue when its job is dispatched to a Linux host using PAM. PAM limits are system resource limits defined in the `limits.conf` file. PAM limits override other limits. If the execution host does not have PAM configured and this parameter is enabled, the job fails.

For parallel jobs, **USE_PAM_CREDS** only takes effect on the first execution host.

Overrides **MEMLIMIT_TYPE=Process**.

Overridden (for CPU limit only) by **LSB_JOB_CPULIMIT=y**.

Overridden (for memory limits only) by **LSB_JOB_MEMLIMIT=y**.

Default

n. **USE_PAM_CREDS** is disabled.

USE_PRIORITY_IN_POOL

Queue-based fairshare only. After job scheduling occurs for each queue, this parameter enables LSF to dispatch jobs to any remaining slots in the pool in first-come first-served order across queues.

Syntax

`USE_PRIORITY_IN_POOL= y | Y | n | N`

Default

N

USERS

Specifies a space-separated list of user names or user groups that can submit jobs to the queue.

Syntax

`USERS=all [~user_name ...] [~user_group ...] | [user_name ...] [user_group [~user_group ...] ...]`

Description

LSF cluster administrators are automatically included in the list of users. LSF cluster administrators can submit jobs to this queue, or switch (**bswitch**) any user's jobs into this queue.

If user groups are specified, each user in the group can submit jobs to this queue. If FAIRSHARE is also defined in this queue, only users defined by both parameters can submit jobs, so LSF administrators cannot use the queue if they are not included in the share assignments.

User names must be valid login names. To specify a Windows user account, include the domain name in uppercase letters (`DOMAIN_NAME\user_name`).

User group names can be LSF user groups or UNIX and Windows user groups. To specify a Windows user group, include the domain name in uppercase letters (`DOMAIN_NAME\user_group`).

Use the keyword `all` to specify all users or user groups in a cluster.

Use the not operator (`~`) to exclude users from the `all` specification or from user groups. This is useful if you have a large number of users but only want to exclude a few users or groups from the queue definition.

The not operator (`~`) can only be used with the `all` keyword or to exclude users from user groups.



CAUTION: The not operator (`~`) does not exclude LSF administrators from the queue definition.

lsb.reasons

Default

all (all users can submit jobs to the queue)

Examples

- `USERS=user1 user2`
- `USERS=all ~user1 ~user2`
- `USERS=all ~ugroup1`
- `USERS=groupA ~user3 ~user4`

Automatic time-based configuration

Variable configuration is used to automatically change LSF configuration based on time windows.

You define automatic configuration changes in `lsb.queues` by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the **badadmin reconfig** command.

The expressions are evaluated by LSF every 10 minutes based on `mbatchd` start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting `mbatchd`, providing continuous system availability.

Example

```
Begin Queue
...
#if time(8:30-18:30 EDT)
INTERACTIVE = ONLY # interactive only during day shift #endif
#endif
...
End Queue
```

Specifying the time zone is optional. If you do not specify a time zone, LSF uses the local system time zone. LSF supports all standard time zone abbreviations.

lsb.reasons

The `lsb.reasons` file allows for individual configuration of pending reason messages. This enables administrators to make messages clear and may inform users on which action they may take to allow the job to run. Messages may be customized for one or more pending reasons as well as priority given to particular resources.

This file is optional. It is installed by default in `config/lsbatch/<cluster_name>/configdir/lsb.reasons`.

After making any changes to `lsb.reasons`, run **badadmin reconfig**

#INCLUDE

Syntax

```
#INCLUDE "path-to-file"
```

Description

Inserts a configuration setting from another file to the current location. Use this directive to dedicate control of a portion of the configuration to other users or user groups by providing write access for the included file to specific users or user groups, and to ensure consistency of configuration file settings in different clusters (if you are using the LSF multicluster capability).

See more information on shared configuration file content in *Administering IBM Spectrum LSF*

#INCLUDE can be inserted anywhere in the local configuration file.

Default

Not defined.

CustomPendingReason section

Configurable message for one or more pending reasons

In the `lsb.reasons` configuration file, each `CustomPendingReason` section may be configured with a message to describe one or more pending reason messages. The pending reason messages are identified with a `REASON_ID` number. All pending reasons and their associated ID numbers are included at the end of the `lsb.reasons` file. Both main reasons and host pending reasons can be specified in the `REASON_ID` list.

For example, the following reasons are related to a dispatched slave host:

```
Begin CustomPendingReason
REASON_ID=1603 1604 1605 1606
MESSAGE=There is something wrong with the allocated host, \
try another with bmod -m 'host (group) list'
End CustomPendingReason
```

In this case, if a job is pending for any of the reasons listed in `REASON_ID`, the user gets a pending reason message telling them to specify another host.

Configurable message and priority for a particular resource

To allow users to highlight key resources, `lsb.reasons` can be used to prioritize the pending reason for particular resources in the cluster and a custom message can be shown when those resources are unavailable. For example, in many cases, software licenses are the most critical or are scarce resources. If a job is pending due to the lack of a license, any other reasons do not matter. However, access to hardware, a specific critical license, or some other resource may be the most relevant reason to the user community. For these resources, it may be preferable to specify something more informative than "Job's requirements for reserving resource (xyz) not satisfied". Resources that are configured with a priority in `lsb.reasons` always have higher priority than those not configured in `lsb.reasons`. If a resource is only configured with a message, it shares the same priority with those resources not configured in `lsb.reasons`. In `lsb.reasons`, at least one of `MESSAGE` or `PRIORITY` must be specified when defining the custom pending reason for a resource. In the following example, three resources (`emu_a`, `lic_a`, and `gpu`) are configured with a customized message and a priority. If a job requested all three resources, and not all of these requirements could be met by a host, the message `<No Type A emulators available>` would be shown for the pending job as `emu_a` has the highest priority.

```
Begin CustomPendingReason
RESOURCE=emu_a
MESSAGE=No type A emulators available
PRIORITY=107
End CustomPendingReason

Begin CustomPendingReason
RESOURCE=lic_a
MESSAGE=No license for SimWorkBench
PRIORITY=10
End CustomPendingReason

Begin CustomPendingReason
RESOURCE=gpu
MESSAGE=All GPUs are in use
PRIORITY=5
End CustomPendingReason
```

Interaction

A custom message for a resource takes priority over a custom message for pending reason.

Support for the LSF multicluster capability

When using the LSF multicluster capability, if both the submission cluster and the execution cluster have a custom pending reason message configured, the message is shown according to the submission cluster configuration. If both the submission cluster and the execution cluster have priority for a resource configured, the priority of the resource in the execution cluster applies to the selection of the single reason.

lsb.resources

The `lsb.resources` file contains configuration information for resource allocation limits, exports, resource usage limits, and guarantee policies. This file is optional.

The `lsb.resources` file is stored in the directory `LSB_CONFDIR/cluster_name/configdir`, where **LSB_CONFDIR** is defined in the `lsf.conf` file.

Changing lsb.resources configuration

After changing the `lsb.resources` file, run **badadmin reconfig** to reconfigure `mbatchd`.

#INCLUDE

Syntax

```
#INCLUDE "path-to-file"
```

Description

Inserts a configuration setting from another file to the current location. Use this directive to dedicate control of a portion of the configuration to other users or user groups by providing write access for the included file to specific users or user groups, and to ensure consistency of configuration file settings in different clusters (if you are using the LSF multicluster capability).

See more information on shared configuration file content in *Administering IBM Spectrum LSF*

`#INCLUDE` can be inserted anywhere in the local configuration file.

Default

Not defined.

Limit section

The Limit section sets limits for the maximum amount of the specified resources that must be available for different classes of jobs to start, and which resource consumers the limits apply to. Limits are enforced during job resource allocation.

Tip:

For limits to be enforced, jobs must specify resource requirements (**bsub -R** or `RES_REQ` in `lsb.queues`).

The **blimits** command displays view current usage of resource allocation limits configured in Limit sections in `lsb.resources`:

Limit section structure

Each set of limits is defined in a Limit section enclosed by `Begin Limit` and `End Limit`.

A Limit section has two formats:

- Vertical tabular

- Horizontal

The file can contain sections in both formats. In either format, you must configure a limit for at least one consumer and one resource. The Limit section cannot be empty.

Vertical tabular format

Use the vertical format for simple configuration conditions involving only a few consumers and resource limits.

The first row consists of an optional NAME and the following keywords for:

- Resource types:
 - Job slots (**SLOTS**) and per-processor job slots (**SLOTS_PER_PROCESSOR**).
 - Memory (**MEM**), in MB or the unit set in the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file.
 - Swap space (**SWP**), in MB or the unit set in the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file.
 - Temp space (**TMP**), in MB or the unit set in the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file.
 - Running and suspended (RUN, SSUSP, USUSP) jobs (**JOBS**).
 - Other shared resources (**RESOURCE**).
- Consumer types:
 - Applications (**APPS** or **PER_APP**).
 - Queues (**QUEUES** or **PER_QUEUE**).
 - Hosts and host groups (**HOSTS** or **PER_HOST**).
 - Users and user groups (**USERS** or **PER_USER**).
 - Projects (**PROJECTS** or **PER_PROJECT**).
 - LSF License Scheduler projects (**LIC_PROJECTS** or **PER_LIC_PROJECT**).

Each subsequent row describes the configuration information for resource consumers and the limits that apply to them. Each line must contain an entry for each keyword. Use empty parentheses () or a dash (-) to indicate an empty field. Fields cannot be left blank.

Tip:

Multiple entries must be enclosed in parentheses. For RESOURCE limits, RESOURCE names must be enclosed in parentheses.

Horizontal format

Use the horizontal format to give a name for your limits and to configure more complicated combinations of consumers and resource limits.

The first line of the Limit section gives the name of the limit configuration.

Each subsequent line in the Limit section consists of keywords identifying the resource limits:

- Job slots (**SLOTS**) and per-processor job slots (**SLOTS_PER_PROCESSOR**).
- Memory (**MEM**), in MB or the unit set in the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file.
- Swap space (**SWP**), in MB or the unit set in the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file.
- Temp space (**TMP**), in MB or the unit set in the **LSF_UNIT_FOR_LIMITS** parameter in the `lsf.conf` file.
- Running and suspended (RUN, SSUSP, USUSP) jobs (**JOBS**).
- Other shared resources (**RESOURCE**).

and the resource *consumers* to which the limits apply:

- Applications (**APPS** or **PER_APP**).
- Queues (**QUEUES** or **PER_QUEUE**).
- Hosts and host groups (**HOSTS** or **PER_HOST**).
- Users and user groups (**USERS** or **PER_USER**).
- Projects (**PROJECTS** or **PER_PROJECT**).
- LSF License Scheduler projects (**LIC_PROJECTS** or **PER_LIC_PROJECT**).

Example: Vertical tabular format

In the following limit configuration:

- Jobs from `user1` and `user3` are limited to 2 job slots on `hostA`
- Jobs from `user2` on queue `normal` are limited to 20 MB of memory or the unit set in **LSF_UNIT_FOR_LIMITS** in `lsf.conf`.
- The short queue can have at most 200 running and suspended jobs

```
Begin Limit
NAME    USERS          QUEUES  HOSTS    SLOTS  MEM  SWP  TMP  JOBS
limit1  (user1 user3)      -       hostA    2      -    -    -    -
-       user2             normal  -        -      20   -    -    -
-       -                 short   -        -      -    -    -    200
End Limit
```

Jobs that do not match these limits; that is, all users except `user1` and `user3` running jobs on `hostA` and all users except `user2` submitting jobs to queue `normal`, have no limits.

Example: Horizontal format

All users in user group `ugroup1` except `user1` using queue1 and queue2 and running jobs on hosts in host group `hgroup1` are limited to 2 job slots per processor on each host:

```
Begin Limit
# ugroup1 except user1 uses queue1 and queue2 with 2 job slots
# on each host in hgroup1
NAME          = limit1
# Resources
SLOTS_PER_PROCESSOR = 2
#Consumers
QUEUES        = queue1 queue2
USERS         = ugroup1 ~user1
PER_HOST      = hgroup1
End Limit
```

Compatibility with `lsb.queues`, `lsb.users`, and `lsb.hosts`

The Limit section does not support the keywords or format used in `lsb.users`, `lsb.hosts`, and `lsb.queues`. However, your existing job slot limit configuration in these files will continue to apply.

Job slot limits are the only type of limit you can configure in `lsb.users`, `lsb.hosts`, and `lsb.queues`. You cannot configure limits for user groups, host groups and projects in `lsb.users`, `lsb.hosts`, and `lsb.queues`. You should not configure any new resource allocation limits in `lsb.users`, `lsb.hosts`, and `lsb.queues`. Use this section to configure all new resource allocation limits, including job slot limits. Limits on running and suspended jobs can only be set in this section.

Existing limits in `lsb.users`, `lsb.hosts`, and `lsb.queues` with the same scope as a new limit in this section, but with a different value are ignored. The value of the new limit in this section is used. Similar limits with different scope enforce the most restrictive limit.

Parameters

- APPS
- ELIGIBLE_PEND_JOBS

- HOSTS
- INELIGIBLE
- JOBS
- JOBS_PER_SCHED_CYCLE
- MEM
- NAME
- PER_APP
- PER_HOST
- PER_PROJECT
- PER_QUEUE
- PER_USER
- PROJECTS
- QUEUES
- RESOURCE
- SLOTS
- SLOTS_PER_PROCESSOR
- SWP
- TMP
- USERS

APPS

Syntax

APPS=all [~]application_profile_name ...

APPS

([-] | all [~]application_profile_name ...)

Description

A space-separated list of application profile names on which limits are enforced. Limits are enforced on all application profiles listed.

The list must contain valid application profile names defined in `lsb.applications`.

To specify a per-application limit, use the `PER_APP` keyword. Do not configure `APPS` and `PER_APP` limits in the same Limit section.

In horizontal format, use only one `APPS` line per Limit section.

Use the keyword `all` to configure limits that apply to all applications in a cluster.

Use the not operator (`~`) to exclude applications from the `all` specification in the limit. This is useful if you have a large number of applications but only want to exclude a few applications from the limit definition.

In vertical tabular format, multiple application profile names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses (`()`) or a dash (`-`) to indicate an empty field. Fields cannot be left blank.

Default

None. If no limit is specified for `PER_APP` or `APPS`, no limit is enforced on any application profile.

Example

```
APPS=appA appB
```

ELIGIBLE_PEND_JOBS**Syntax**

```
ELIGIBLE_PEND_JOBS=integer
```

Description

The maximum number of eligible jobs that are considered for dispatch in a single scheduling cycle. Specify a positive integer or 0. This parameter can only be used with the following consumer types:

- USERS or PER_USER
- QUEUES or PER_QUEUE

The **all** group, or any group containing **all**, is not supported.

The limit is ignored for any other defined consumer types.

Valid values

Any positive integer or 0.

Default

No limit

Example

```
ELIGIBLE_PEND_JOBS=10
```

HOSTS**Syntax**

```
HOSTS=all [~]host_name ... | all [~]host_group ...
```

```
HOSTS
```

```
( [-] | all [~]host_name ... | all [~]host_group ... )
```

Description

A space-separated list of hosts, host groups defined in `lsb.hosts` on which limits are enforced. Limits are enforced on all hosts or host groups listed.

If a group contains a subgroup, the limit also applies to each member in the subgroup recursively.

To specify a per-host limit, use the PER_HOST keyword. Do not configure HOSTS and PER_HOST limits in the same Limit section.

If you specify MEM, TMP, or SWP as a percentage, you must specify PER_HOST and list the hosts that the limit is to be enforced on. You cannot specify HOSTS.

In horizontal format, use only one HOSTS line per Limit section.

Use the keyword all to configure limits that apply to all hosts in a cluster.

Use the not operator (~) to exclude hosts from the all specification in the limit. This is useful if you have a large cluster but only want to exclude a few hosts from the limit definition.

In vertical tabular format, multiple host names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate an empty field. Fields cannot be left blank.

Default

all (limits are enforced on all hosts in the cluster).

Example 1

```
HOSTS=Group1 ~hostA hostB hostC
```

Enforces limits on hostB, hostC, and all hosts in Group1 except for hostA.

Example 2

```
HOSTS=all ~group2 ~hostA
```

Enforces limits on all hosts in the cluster, except for hostA and the hosts in group2.

Example 3

```
HOSTS                                SWP (all ~hostK ~hostM)          10
```

Enforces a 10 MB (or the unit set in **LSF_UNIT_FOR_LIMITS** in `lsf.conf`) swap limit on all hosts in the cluster, except for hostK and hostM

INELIGIBLE

Syntax

```
INELIGIBLE=Y|y|N|n
```

Description

If set as y or Y in the specific Limit section and the job cannot be scheduled due to this limit, the LSF scheduler will put the job into an ineligible pending state. LSF calculates the ineligible pending time for this job and the job's priority does not increase.

Note: The following Limit types are compatible with the INELIGIBLE parameter: JOBS, USERS, PER_USER, QUEUES, PER_QUEUE, PROJECTS, PER_PROJECT, LIC_PROJECTS, and PER_LIC_PROJECT.

Default

```
INELIGIBLE=N
```

Example

```
INELIGIBLE=Y
```

JOBS

Syntax

```
JOBS=integer
```

```
JOBS
```

```
- | integer
```

Description

Maximum number of running or suspended (RUN, SSUSP, USUSP) jobs available to resource consumers. Specify a positive integer greater than or equal 0. Job limits can be defined in both vertical and horizontal limit formats.

With MultiCluster resource lease model, this limit applies only to local hosts being used by the local cluster. The job limit for hosts exported to a remote cluster is determined by the host export policy, not by this parameter. The job limit for borrowed hosts is determined by the host export policy of the remote cluster.

If SLOTS are configured in the Limit section, the most restrictive limit is applied.

If HOSTS are configured in the Limit section, JOBS is the number of running and suspended jobs on a host. If preemptive scheduling is used, the suspended jobs are not counted against the job limit.

Use this parameter to prevent a host from being overloaded with too many jobs, and to maximize the throughput of a machine.

If only QUEUES are configured in the Limit section, JOBS is the maximum number of jobs that can run in the listed queues.

If only USERS are configured in the Limit section, JOBS is the maximum number of jobs that the users or user groups can run.

If only HOSTS are configured in the Limit section, JOBS is the maximum number of jobs that can run on the listed hosts.

If only PROJECTS are configured in the Limit section, JOBS is the maximum number of jobs that can run under the listed projects.

Use APPS or PER_APP, QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, LIC_PROJECTS or PER_LIC_PROJECT, and PROJECTS or PER_PROJECT in combination to further limit jobs available to resource consumers.

In horizontal format, use only one JOBS line per Limit section.

In vertical format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

Default

No limit

Example

```
JOBS=20
```

JOBS_PER_SCHED_CYCLE

Syntax

```
JOBS_PER_SCHED_CYCLE=integer
```

Description

Use **ELIGIBLE_PEND_JOBS** instead. This parameter is only maintained for backwards compatibility.

Maximum number of jobs that are considered for dispatch in a single scheduling cycle. Specify a positive integer or 0. This parameter can only be used with the following consumer types:

- USERS or PER_USER
- QUEUES or PER_QUEUE

The **all** group, or any group containing **all**, is not supported.

The limit is ignored for any other defined consumer types.

Valid values

Any positive integer or 0.

Default

No limit

Example

```
JOBS_PER_SCHED_CYCLE=10
```

LIC_PROJECTS

Syntax

```
LIC_PROJECTS=all [~]lic_project_name ...
```

```
LIC_PROJECTS
```

```
( [-] | all [~]lic_project_name ... )
```

Description

A space-separated list of LSF License Scheduler project names on which limits are enforced. Limits are enforced on all projects listed. Each project name can be up to 511 characters long.

To specify a per-project limit, use the PER_LIC_PROJECT keyword. Do not configure LIC_PROJECTS and PER_PROJECT limits in the same Limit section.

In horizontal format, use only one LIC_PROJECTS line per Limit section.

Use the keyword all to configure limits that apply to all projects in a cluster.

Use the not operator (~) to exclude projects from the all specification in the limit. This is useful if you have a large number of projects but only want to exclude a few projects from the limit definition.

In vertical tabular format, multiple project names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate an empty field. Fields cannot be left blank.

Default

all (limits are enforced on all projects in the cluster)

Example

```
LIC_PROJECTS=licprojA licprojB
```

MEM

Syntax

```
MEM=integer[%]
```

```
MEM
```

```
- | integer[%]
```

Description

Maximum amount of memory available to resource consumers. Specify a value in MB or the unit set in **LSF_UNIT_FOR_LIMITS** in `lsf.conf` as a positive integer greater than or equal 0.

The Limit section is ignored if MEM is specified as a percentage:

- Without PER_HOST, or
- With HOSTS

In horizontal format, use only one MEM line per Limit section.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

If only QUEUES are configured in the Limit section, MEM must be an integer value. MEM is the maximum amount of memory available to the listed queues.

If only USERS are configured in the Limit section, MEM must be an integer value. MEM is the maximum amount of memory that the users or user groups can use.

If only HOSTS are configured in the Limit section, MEM must be an integer value. It cannot be a percentage. MEM is the maximum amount of memory available to the listed hosts.

If only PROJECTS are configured in the Limit section, MEM must be an integer value. MEM is the maximum amount of memory available to the listed projects.

Use APPS or PER_APP, QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, LIC_PROJECTS or PER_LIC_PROJECT, and PROJECTS or PER_PROJECT in combination to further limit memory available to resource consumers.

Default

No limit

Example

```
MEM=20
```

NAME**Syntax**

NAME=*limit_name*

NAME

- | *limit_name*

Description

Name of the Limit section

Specify any ASCII string 40 characters or less. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces.

If duplicate limit names are defined, the Limit section is ignored. If value of NAME is not defined in vertical format, or defined as (-), **blimitis** displays NONAME*nnn*.

Default

None. In horizontal format, you must provide a name for the Limit section. NAME is optional in the vertical format.

Example

```
NAME=short_limits
```

PER_APP**Syntax**

```
PER_APP=all [~]application_profile_name ..
```

```
PER_APP
```

```
( [-] | all [~]application_profile_name ... )
```

Description

A space-separated list of application profile names on which limits are enforced. Limits are enforced on jobs submitted to each application profile listed.

Do not configure PER_APP and APPS limits in the same Limit section.

In horizontal format, use only one PER_APP line per Limit section.

Use the keyword all to configure limits that apply to each application in a cluster.

Use the not operator (~) to exclude applications from the all specification in the limit.

In vertical tabular format, multiple application profile names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate an empty field. Fields cannot be left blank.

Default

None. If no limit is specified for PER_APP or APPS, no limit is enforced on any application profile.

Example

```
PER_APP=app1 app2
```

PER_HOST**Syntax**

```
PER_HOST=all [~]host_name ... | all [~]host_group ...
```

```
PER_HOST
```

```
( [-] | all [~]host_name ... | all [~]host_group ... )
```

Description

A space-separated list of host or host groups defined in `lsb.hosts` on which limits are enforced. Limits are enforced on each host or individually to each host of the host group listed. If a group contains a subgroup, the limit also applies to each member in the subgroup recursively.

Do not configure PER_HOST and HOSTS limits in the same Limit section.

In horizontal format, use only one PER_HOST line per Limit section.

If you specify MEM, TMP, or SWP as a percentage, you must specify PER_HOST and list the hosts that the limit is to be enforced on. You cannot specify HOSTS.

Use the keyword all to configure limits that apply to each host in a cluster. If host groups are configured, the limit applies to each member of the host group, not the group as a whole.

Use the not operator (~) to exclude hosts or host groups from the all specification in the limit. This is useful if you have a large cluster but only want to exclude a few hosts from the limit definition.

In vertical tabular format, multiple host names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate an empty field. Fields cannot be left blank.

Default

None. If no limit is specified for PER_HOST or HOST, no limit is enforced on any host or host group.

Example

```
PER_HOST=hostA hgroup1 ~hostC
```

PER_LIC_PROJECT**Syntax**

```
PER_LIC_PROJECT=all [~]lic_project_name ...
```

```
PER_LIC_PROJECT
```

```
( [-] | all [~]lic_project_name ... )
```

Description

A space-separated list of LSF License Scheduler project names on which limits are enforced. Limits are enforced on each project listed. Each project name can be up to 511 characters long.

Do not configure PER_LIC_PROJECT and LIC_PROJECTS limits in the same Limit section.

In horizontal format, use only one PER_LIC_PROJECT line per Limit section.

Use the keyword all to configure limits that apply to each LSF License Scheduler project in a cluster.

Use the not operator (~) to exclude LSF License Scheduler projects from the all specification in the limit.

In vertical tabular format, multiple project names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate an empty field. Fields cannot be left blank.

Default

None. If no limit is specified for PER_LIC_PROJECT or LIC_PROJECTS, no limit is enforced on any LSF License Scheduler project.

Example

```
PER_PROJECT=licproj1 licproj2
```

PER_PROJECT**Syntax**

```
PER_PROJECT=all [~]project_name ...
```

```
PER_PROJECT
```

```
( [-] | all [~]project_name ... )
```

Description

A space-separated list of project names on which limits are enforced. Limits are enforced on each project listed. Each project name can be up to 511 characters long.

Do not configure PER_PROJECT and PROJECTS limits in the same Limit section.

In horizontal format, use only one PER_PROJECT line per Limit section.

Use the keyword all to configure limits that apply to each project in a cluster.

Use the not operator (~) to exclude projects from the all specification in the limit.

In vertical tabular format, multiple project names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate an empty field. Fields cannot be left blank.

Default

None. If no limit is specified for PER_PROJECT or PROJECTS, no limit is enforced on any project.

Example

```
PER_PROJECT=proj1 proj2
```

PER_QUEUE**Syntax**

```
PER_QUEUE=all [~]queue_name ..
```

```
PER_QUEUE
```

```
( [-] | all [~]queue_name ... )
```

Description

A space-separated list of queue names on which limits are enforced. Limits are enforced on jobs submitted to each queue listed.

Do not configure PER_QUEUE and QUEUES limits in the same Limit section.

In horizontal format, use only one PER_QUEUE line per Limit section.

Use the keyword all to configure limits that apply to each queue in a cluster.

Use the not operator (~) to exclude queues from the all specification in the limit. This is useful if you have a large number of queues but only want to exclude a few queues from the limit definition.

In vertical tabular format, multiple queue names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate an empty field. Fields cannot be left blank.

Default

None. If no limit is specified for PER_QUEUE or QUEUES, no limit is enforced on any queue.

Example

```
PER_QUEUE=priority night
```

PER_USER**Syntax**

```
PER_USER=all [~]user_name ... | all [~]user_group ...
```

```
PER_USER
```

```
( [-] | all [~]user_name ... | all [~]user_group ... )
```

Description

A space-separated list of user names or user groups on which limits are enforced. Limits are enforced on each user or individually to each user in the user group listed. If a user group contains a subgroup, the limit also applies to each member in the subgroup recursively.

User names must be valid login names. User group names can be LSF user groups or UNIX and Windows user groups. Note that for LSF and UNIX user groups, the groups must be specified in a UserGroup section in `lsb.users` first.

Do not configure PER_USER and USERS limits in the same Limit section.

In horizontal format, use only one PER_USER line per Limit section.

Use the keyword `all` to configure limits that apply to each user in a cluster. If user groups are configured, the limit applies to each member of the user group, not the group as a whole.

Use the not operator (`~`) to exclude users or user groups from the `all` specification in the limit. This is useful if you have a large number of users but only want to exclude a few users from the limit definition.

In vertical tabular format, multiple user names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses (`()`) or a dash (`-`) to indicate an empty field. Fields cannot be left blank.

Default

None. If no limit is specified for PER_USER or USERS, no limit is enforced on any user or user group.

Example

```
PER_USER=user1 user2 ugroup1 ~user3
```

PROJECTS**Syntax**

```
PROJECTS=all [~]project_name ...
```

```
PROJECTS
```

```
( [-] | all [~]project_name ... )
```

Description

A space-separated list of project names on which limits are enforced. Limits are enforced on all projects listed. Each project name can be up to 511 characters long.

To specify a per-project limit, use the PER_PROJECT keyword. Do not configure PROJECTS and PER_PROJECT limits in the same Limit section.

In horizontal format, use only one PROJECTS line per Limit section.

Use the keyword `all` to configure limits that apply to all projects in a cluster.

Use the not operator (~) to exclude projects from the all specification in the limit. This is useful if you have a large number of projects but only want to exclude a few projects from the limit definition.

In vertical tabular format, multiple project names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate an empty field. Fields cannot be left blank.

Default

all (limits are enforced on all projects in the cluster)

Example

```
PROJECTS=projA projB
```

QUEUES

Syntax

```
QUEUES=all [~]queue_name ...
```

```
QUEUES
```

```
( [-] | all [~]queue_name ... )
```

Description

A space-separated list of queue names on which limits are enforced. Limits are enforced on all queues listed.

The list must contain valid queue names defined in `lsb.queues`.

To specify a per-queue limit, use the `PER_QUEUE` keyword. Do not configure `QUEUES` and `PER_QUEUE` limits in the same Limit section.

In horizontal format, use only one `QUEUES` line per Limit section.

Use the keyword `all` to configure limits that apply to all queues in a cluster.

Use the not operator (~) to exclude queues from the all specification in the limit. This is useful if you have a large number of queues but only want to exclude a few queues from the limit definition.

In vertical tabular format, multiple queue names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate an empty field. Fields cannot be left blank.

Default

all (limits are enforced on all queues in the cluster)

Example

```
QUEUES=normal night
```

RESOURCE

Syntax

```
RESOURCE=[shared_resource, integer] [[shared_resource, integer] ...]
```

```
RESOURCE
```

```
( [[shared_resource, integer] [[shared_resource, integer] ...] )
```

Description

Maximum amount of any user-defined shared resource available to consumers.

In horizontal format, use only one RESOURCE line per Limit section.

In vertical tabular format, resource names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate an empty field. Fields cannot be left blank.

Default

None

Examples

```
RESOURCE=[stat_shared,4]
Begin Limit
RESOURCE                PER_HOST
([stat_shared,4])       (all ~hostA)
([dyn_rsrc,1] [stat_rsrc,2]) (hostA)
End Limit
```

SLOTS**Syntax**

SLOTS=*integer*

SLOTS

- | *integer*

Description

Maximum number of job slots available to resource consumers. Specify a positive integer greater than or equal 0.

With MultiCluster resource lease model, this limit applies only to local hosts being used by the local cluster. The job slot limit for hosts exported to a remote cluster is determined by the host export policy, not by this parameter. The job slot limit for borrowed hosts is determined by the host export policy of the remote cluster.

If JOBS are configured in the Limit section, the most restrictive limit is applied.

If HOSTS are configured in the Limit section, SLOTS is the number of running and suspended jobs on a host. If preemptive scheduling is used, the suspended jobs are not counted as using a job slot.

To fully use the CPU resource on multiprocessor hosts, make the number of job slots equal to or greater than the number of processors.

Use this parameter to prevent a host from being overloaded with too many jobs, and to maximize the throughput of a machine.

Use an exclamation point (!) to make the number of job slots equal to the number of CPUs on a host.

If the number of CPUs in a host changes dynamically, **mbatchd** adjusts the maximum number of job slots per host accordingly. Allow the **mbatchd** up to 10 minutes to get the number of CPUs for a host. During this period the value of SLOTS is 1.

If only QUEUES are configured in the Limit section, SLOTS is the maximum number of job slots available to the listed queues.

If only USERS are configured in the Limit section, SLOTS is the maximum number of job slots that the users or user groups can use.

If only HOSTS are configured in the Limit section, SLOTS is the maximum number of job slots that are available to the listed hosts.

If only PROJECTS are configured in the Limit section, SLOTS is the maximum number of job slots that are available to the listed projects.

Use APPS or PER_APP, QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, LIC_PROJECTS or PER_LIC_PROJECT, and PROJECTS or PER_PROJECT in combination to further limit job slots per processor available to resource consumers.

In horizontal format, use only one SLOTS line per Limit section.

In vertical format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

Default

No limit

Example

```
SLOTS=20
```

SLOTS_PER_PROCESSOR

Syntax

```
SLOTS_PER_PROCESSOR=number
```

```
SLOTS_PER_PROCESSOR
```

```
- | number
```

Description

Per processor job slot limit, based on the number of processors on each host affected by the limit.

Maximum number of job slots that each resource consumer can use per processor. This job slot limit is configured per processor so that multiprocessor hosts will automatically run more jobs.

You must also specify PER_HOST and list the hosts that the limit is to be enforced on. The Limit section is ignored if SLOTS_PER_PROCESSOR is specified:

- Without PER_HOST, or
- With HOSTS

In vertical format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

To fully use the CPU resource on multiprocessor hosts, make the number of job slots equal to or greater than the number of processors.

Use this parameter to prevent a host from being overloaded with too many jobs, and to maximize the throughput of a machine.

This number can be a fraction such as 0.5, so that it can also serve as a per-CPU limit on multiprocessor machines. This number is rounded up to the nearest integer equal to or greater than the total job slot limits for a host. For example, if SLOTS_PER_PROCESSOR is 0.5, on a 4-CPU multiprocessor host, users can only use up to 2 job slots at any time. On a single-processor machine, users can use 1 job slot.

Use an exclamation point (!) to make the number of job slots equal to the number of CPUs on a host.

If the number of CPUs in a host changes dynamically, mbatchd adjusts the maximum number of job slots per host accordingly. Allow the mbatchd up to 10 minutes to get the number of CPUs for a host. During this period the number of CPUs is 1.

If only QUEUES and PER_HOST are configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor available to the listed queues for any hosts, users or projects.

If only USERS and PER_HOST are configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor that the users or user groups can use on any hosts, queues, license projects, or projects.

If only PER_HOST is configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor available to the listed hosts for any users, queues or projects.

If only PROJECTS and PER_HOST are configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor available to the listed projects for any users, queues or hosts.

Use APPS or PER_APP, QUEUES or PER_QUEUE, USERS or PER_USER, PER_HOST, LIC_PROJECTS or PER_LIC_PROJECT, and PROJECTS or PER_PROJECT in combination to further limit job slots per processor available to resource consumers.

Default

No limit

Example

```
SLOTS_PER_PROCESSOR=2
```

SWP

Syntax

SWP=*integer*[%]

SWP

- | *integer*[%]

Description

Maximum amount of swap space available to resource consumers. Specify a value in MB or the unit set in **LSF_UNIT_FOR_LIMITS** in `lsf.conf` as a positive integer greater than or equal 0.

The Limit section is ignored if SWP is specified as a percentage:

- Without PER_HOST, or
- With HOSTS

In horizontal format, use only one SWP line per Limit section.

In vertical format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

If only USERS are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space that the users or user groups can use on any hosts, queues or projects.

If only HOSTS are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space available to the listed hosts for any users, queues or projects.

If only PROJECTS are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space available to the listed projects for any users, queues or hosts.

If only LIC_PROJECTS are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space available to the listed projects for any users, queues, projects, or hosts.

Use APPS or PER_APP, QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, LIC_PROJECTS or PER_LIC_PROJECT, and PROJECTS or PER_PROJECT in combination to further limit swap space available to resource consumers.

Default

No limit

Example

SWP=60

TMP**Syntax**TMP=*integer*[%]

TMP

- | *integer*[%]**Description**

Maximum amount of tmp space available to resource consumers. Specify a value in MB or the unit set in **LSF_UNIT_FOR_LIMITS** in `lsf.conf` as a positive integer greater than or equal 0.

The Limit section is ignored if TMP is specified as a percentage:

- Without PER_HOST, or
- With HOSTS

In horizontal format, use only one TMP line per Limit section.

In vertical format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

If only QUEUES are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of tmp space available to the listed queues for any hosts, users projects.

If only USERS are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of tmp space that the users or user groups can use on any hosts, queues or projects.

If only HOSTS are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of tmp space available to the listed hosts for any users, queues or projects.

If only PROJECTS are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of tmp space available to the listed projects for any users, queues or hosts.

If only LIC_PROJECTS are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of tmp space available to the listed projects for any users, queues, projects, or hosts.

Use APPS or PER_APP, QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, LIC_PROJECTS or PER_LIC_PROJECT, and PROJECTS or PER_PROJECT in combination to further limit tmp space available to resource consumers.

Default

No limit

Example

TMP=20%

USERS**Syntax**

```
USERS=all [~]user_name ... | all [~]user_group ...
```

```
USERS
```

```
( [-] | all [~]user_name ... | all [~]user_group ... )
```

Description

A space-separated list of user names or user groups on which limits are enforced. Limits are enforced on all users or groups listed. Limits apply to a group as a whole.

If a group contains a subgroup, the limit also applies to each member in the subgroup recursively.

User names must be valid login names. User group names can be LSF user groups or UNIX and Windows user groups. UNIX user groups must be configured in `lsb.user`.

To specify a per-user limit, use the `PER_USER` keyword. Do not configure `USERS` and `PER_USER` limits in the same Limit section.

In horizontal format, use only one `USERS` line per Limit section.

Use the keyword `all` to configure limits that apply to all users or user groups in a cluster.

Use the not operator (`~`) to exclude users or user groups from the all specification in the limit. This is useful if you have a large number of users but only want to exclude a few users or groups from the limit definition.

In vertical format, multiple user names must be enclosed in parentheses.

In vertical format, use empty parentheses (`()`) or a dash (`-`) to indicate an empty field. Fields cannot be left blank.

Default

all (limits are enforced on all users in the cluster)

Example

```
USERS=user1 user2
```

GuaranteedResourcePool section

Defines a guarantee policy. A guarantee is a commitment to ensure availability of a number of resources to a service class, where a service class is a container for jobs. Each guarantee pool can provide guarantees to multiple service classes, and each service class can have guarantees in multiple pools.

To use guaranteed resources, configure service classes with `GOALS=[GUARANTEE]` in the `lsb.serviceclasses` file.

Note: Hosts that are not ready for dispatched jobs are not assigned to the guaranteed resource pool. This include hosts that are in `unavail` or `unreach` status, or hosts that are closed by the administrator.

GuaranteedResourcePool section structure

Each resource pool is defined in a `GuaranteedResourcePool` section and enclosed by `Begin GuaranteedResourcePool` and `End GuaranteedResourcePool`.

You must configure a **NAME**, **TYPE** and **DISTRIBUTION** for each `GuaranteedResourcePool` section.

The order of `GuaranteedResourcePool` sections is important, as the sections are evaluated in the order configured. Each host can only be in one pool of host-based resources (slots, hosts, or package, each of

which can have its own GuaranteedResourcePool section); ensure all GuaranteedResourcePool sections (except the last one) define the **HOSTS** parameter, so they do not contain the default of all hosts.

When LSF starts up, it goes through the hosts and assigns each host to a pool that will accept the host based on the pool's **RES_SELECT** and **HOSTS** parameters. If multiple pools will accept the host, the host will be assigned to the first pool according to the configuration order of the pools.

Example GuaranteedResourcePool sections

```
Begin GuaranteedResourcePool
NAME = linuxGuarantee
TYPE = slots
HOSTS = linux_group
DISTRIBUTION = [sc1, 25] [sc2, 30]
LOAN_POLICIES=QUEUES[all] DURATION[15]
DESCRIPTION = This is the resource pool for the hostgroup linux_group, with\
30 slots guaranteed to sc2. Resources are loaned to 25 slots guaranteed to\
sc1 jobs from any queue with run times of up to 15 minutes.
End GuaranteedResourcePool
```

```
Begin GuaranteedResourcePool
NAME = x86Guarantee
TYPE = slots
HOSTS = linux_x86
DISTRIBUTION = [sc1, 25]
LOAN_POLICIES=QUEUES[short_jobs] DURATION[15]
DESCRIPTION = This is the resource pool for the hostgroup\
linux_x86 using the short_jobs queue, with 25 slots guaranteed\
to sc1. Resources are loaned to jobs for up to 15 minutes.
End GuaranteedResourcePool
```

```
Begin GuaranteedResourcePool
NAME = resource2pool
TYPE = resource[f2]
DISTRIBUTION = [sc1, 25%] [sc2, 25%]
LOAN_POLICIES=QUEUES[all] DURATION[10]
DESCRIPTION = This is the resource pool for all f2 resources managed by\
LSF License Scheduler, with 25% guaranteed to each of sc1 and sc2. \
Resources are loaned to jobs from any queue with runtimes of up to 10 minutes.
End GuaranteedResourcePool
```

Parameters

- NAME
- TYPE
- HOSTS
- RES_SELECT
- DISTRIBUTION
- LOAN_POLICIES
- DESCRIPTION
- ADMINISTRATORS

NAME

Syntax

NAME=*name*

Description

The name of the guarantee policy.

Default

None. You must provide a name for the guarantee.

TYPE**Syntax**

```
TYPE = slots | hosts | resource[shared_resource] | package[slots=slots_per_package
[:mem=mem_per_package]]
```

Description

Defines the type of resources to be guaranteed in this guarantee policy. These can either be slots, whole hosts, packages composed of an amount of slots and memory bundled on a single host, or licenses managed by LSF License Scheduler.

Specify `resource[license]` to guarantee licenses (which must be managed by LSF License Scheduler) to service class guarantee jobs.

Specifies the combination of memory and slots that defines the packages that will be treated as resources reserved by service class guarantee jobs. For example,

```
TYPE=package[slots=1:mem=1000]
```

Each unit guaranteed is for one slot and 1000 MB of memory.

LSF_UNIT_FOR_LIMITS in `lsf.conf` determines the units of memory in the package definition. The default value of **LSF_UNIT_FOR_LIMITS** is MB, therefore the guarantee is for 1000 MB of memory.

A package need not have both slots and memory. Setting `TYPE=package[slots=1]` is the equivalent of slots. In order to provide guarantees for parallel jobs that require multiple CPUs on a single host where memory is not an important resource, you can use packages with multiple slots and not specify mem.

Each host can belong to at most one slot/host/package guarantee pool.

Default

None. You must specify the type of guarantee.

HOSTS**Syntax**

```
HOSTS=all | allremote | all@cluster_name ... | [~]host_name | [~]host_group
```

Description

A space-separated list of hosts or host groups defined in `lsb.hosts`, on which the guarantee is enforced.

Use the keyword `all` to include all hosts in a cluster. Use the not operator (`~`) to exclude hosts from the all specification in the guarantee.

Use host groups for greater flexibility, since host groups have additional configuration options.

Ensure all `GuaranteedResourcePool` sections (except the last one) define the **HOSTS** or **RES_SELECT** parameter, so they do not contain the default of all hosts.

Default

all

RES_SELECT

Syntax

`RES_SELECT=res_req`

Description

Resource requirement string with which all hosts defined by the HOSTS parameter are further filtered. For example, `RES_SELECT=type==LINUX86`

Only static host attributes can be used in **RES_SELECT**. Do not use consumable resources or dynamic resources.

Default

None. **RES_SELECT** is optional.

DISTRIBUTION

Syntax

`DISTRIBUTION=([service_class_name, amount[%]]...)`

Description

Assigns the amount of resources in the pool to the specified service classes, where *amount* can be an absolute number or a percentage of the resources in the pool. The outer brackets are optional.

When configured as a percentage, the total can exceed 100% but each assigned percentage cannot exceed 100%. For example:

`DISTRIBUTION=[sc1, 50%] [sc2, 50%] [sc3, 50%]` is an acceptable configuration even though the total percentages assigned add up to 150%.

`DISTRIBUTION=[sc1, 120%]` is not an acceptable configuration, since the percentage for sc1 is greater than 100%.

Each service class must be configured in `lsb.serviceclasses`, with `GOALS=[GUARANTEE]`.

When configured as a percentage and there are remaining resources to distribute (because the calculated number of slots is rounded down), LSF distributes the remaining resources using round-robin distribution, starting with the first configured service class. Therefore, the service classes that you define first will receive additional resources regardless of the configured percentage. For example, there are 93 slots in a pool and you configure the following guarantee distribution:

`DISTRIBUTION=[sc1, 30%] [sc2, 10%] [sc3, 30%]`

The number of slots assigned to guarantee policy are: $\text{floor}((30\% + 10\% + 30\%)*(93 \text{ slots})) = 65 \text{ slots}$

The slots are distributed to the service classes as follows:

- $\text{sc1_slots} = \text{floor}(30\%*92) = 27$
- $\text{sc2_slots} = \text{floor}(10\%*92) = 9$
- $\text{sc3_slots} = \text{floor}(30\%*92) = 27$

As a result of rounding down, the total number of distributed slots is $27+9+27=63$ slots, which means there are two remaining slots to distribute. Using round-robin distribution, LSF distributes one slot each to sc1 and sc2 because these service classes are defined first. Therefore, the final slot distribution to the service classes are as follows:

- $\text{sc1_slots} = \text{floor}(30\%*92) + 1 = 28$
- $\text{sc2_slots} = \text{floor}(10\%*92) + 1 = 10$
- $\text{sc3_slots} = \text{floor}(30\%*92) = 27$

If you configure sc3 before sc2 (DISTRIBUTION=[sc1,30%] [sc3,30%] [sc2,10%]), LSF distributes the two remaining slots to sc1 and sc3. Therefore, the slots are distributed as follows:

- sc1_slots = floor(30%*92) + 1 = 28
- sc3_slots = floor(30%*92) + 1 = 28
- sc2_slots = floor(10%*92) = 9

Default

None. You must provide a distribution for the resource pool.

LOAN_POLICIES**Syntax**

```
LOAN_POLICIES=QUEUES[(!)queue_name ... | a11] [CLOSE_ON_DEMAND] [DURATION[minutes]
[RETAIN[amount%]]]
```

Description

By default, LSF will reserve sufficient resources in each guarantee pool to honor the configured guarantees. To increase utilization, use **LOAN_POLICIES** to allow any job (with or without guarantees) to use these reserved resources when not needed by jobs with guarantees. When resources are loaned out, jobs with guarantees may have to wait for jobs to finish before they are able to dispatch in the pool.

QUEUES[a11 | (!)queue_name ...] loans only to jobs from the specified queue or queues. You must specify which queues are permitted to borrow resources reserved for guarantees. Specify an exclamation point (!) before the queue name for that queue to ignore any **RETAIN** and **DURATION** policies when deciding whether a job in the queue can borrow unused guaranteed resources.

When CLOSE_ON_DEMAND is specified, LSF stops loaning out from a pool whenever there is pending demand from jobs with guarantees in the pool.

DURATION[minutes] only allows jobs to borrow the resources if the job run limit (or estimated run time) is no larger than minutes. Loans limited by job duration make the guaranteed resources available within the time specified by minutes. Jobs running longer than the estimated run time will run to completion regardless of the actual run time.

RETAIN[amount%] enables LSF to try to keep idle the amount of resources specified in **RETAIN** as long as there are unused guarantees. These idle resources can only be used to honor guarantees. Whenever the number of free resources in the pool drops below the **RETAIN** amount, LSF stops loaning resources from the pool.

Default

None. **LOAN_POLICIES** is optional.

DESCRIPTION**Syntax**

```
DESCRIPTION=description
```

Description

A description of the guarantee policy.

Default

None. **DESCRIPTION** is optional.

ADMINISTRATORS

Syntax

ADMINISTRATORS=*user_name* / *user_group* ...

Description

When defined by the user, an administrator can manage the corresponding **GuaranteedResourcePool** by using the **bconf** command.

Note: To specify a Windows user account or user group, include the domain name in uppercase letters: DOMAIN_NAME*user_name* or DOMAIN_NAME*user_group*.

An LSF administrator will also be able to manage a **GuaranteedResourcePool** by using the **bconf** command even if the LSF administrator is excluded in the *user_group* explicitly by using NOT operator (~).

Default

None. You must be a cluster administrator to operate on this guaranteed resource pool by **bconf**.

HostExport section

Defines an export policy for a host or a group of related hosts. Defines how much of each host's resources are exported, and how the resources are distributed among the consumers.

Each export policy is defined in a separate HostExport section, so it is normal to have multiple HostExport sections in `lsb.resources`.

HostExport section structure

Use empty parentheses () or a dash (-) to specify the default value for an entry. Fields cannot be left blank.

Example HostExport section

```
Begin HostExport PER_HOST= hostA hostB SLOTS= 4 DISTRIBUTION= [cluster1, 1]
[cluster2, 3] MEM= 100 SWP= 100 End HostExport
```

Parameters

- PER_HOST
- RES_SELECT
- NHOSTS
- DISTRIBUTION
- MEM
- SLOTS
- SWAP
- TYPE

PER_HOST

Syntax

[“slave.config” on page 658](#)

PER_HOST=*host_name*...

Description

Required when exporting special hosts.

Determines which hosts to export. Specify one or more LSF hosts by name. Separate names by space.

RES_SELECT**Syntax**

RES_SELECT=*res_req*

Description

Required when exporting workstations.

Determines which hosts to export. Specify the selection part of the resource requirement string (without quotes or parentheses), and LSF will automatically select hosts that meet the specified criteria. For this parameter, if you do not specify the required host type, the default is `type==any`.

When LSF_STRICT_RESREQ=Y is configured in `lsf.conf`, resource requirement strings in select sections must conform to a more strict syntax. The strict resource requirement syntax only applies to the select section. It does not apply to the other resource requirement sections (order, rusage, same, span, or cu). When LSF_STRICT_RESREQ=Y in `lsf.conf`, LSF rejects resource requirement strings where an rusage section contains a non-consumable resource.

The criteria is only evaluated once, when a host is exported.

NHOSTS**Syntax**

NHOSTS=*integer*

Description

Required when exporting workstations.

Maximum number of hosts to export. If there are not this many hosts meeting the selection criteria, LSF exports as many as it can.

DISTRIBUTION**Syntax**

DISTRIBUTION=(*[cluster_name, number_shares]...*)

Description

Required. Specifies how the exported resources are distributed among consumer clusters.

The syntax for the distribution list is a series of share assignments. The syntax of each share assignment is the cluster name, a comma, and the number of shares, all enclosed in square brackets, as shown. Use a space to separate multiple share assignments. Enclose the full distribution list in a set of round brackets.

cluster_name

Specify the name of a remote cluster that will be allowed to use the exported resources. If you specify a local cluster, the assignment is ignored.

number_shares

Specify a positive integer representing the number of shares of exported resources assigned to the cluster.

The number of shares assigned to a cluster is only meaningful when you compare it to the number assigned to other clusters, or to the total number. The total number of shares is just the sum of all the shares assigned in each share assignment.

MEM

Syntax

MEM=*megabytes*

Description

Used when exporting special hosts. Specify the amount of memory to export on each host, in MB or in units set in **LSF_UNIT_FOR_LIMITS** in `lsf.conf`.

Default

- (provider and consumer clusters compete for available memory)

SLOTS

Syntax

SLOTS=*integer*

Description

Required when exporting special hosts. Specify the number of job slots to export on each host.

To avoid overloading a partially exported host, you can reduce the number of job slots in the configuration of the local cluster.

SWAP

Syntax

SWAP=*megabytes*

Description

Used when exporting special hosts. Specify the amount of swap space to export on each host, in MB or in units set in **LSF_UNIT_FOR_LIMITS** in `lsf.conf`.

Default

- (provider and consumer clusters compete for available swap space)

TYPE

Syntax

TYPE=*shared*

Description

Changes the lease type from exclusive to shared.

If you export special hosts with a shared lease (using `PER_HOST`), you cannot specify multiple consumer clusters in the distribution policy.

Default

Undefined (the lease type is exclusive; exported resources are never available to the provider cluster)

SharedResourceExport section

Optional. Requires HostExport section. Defines an export policy for a shared resource. Defines how much of the shared resource is exported, and the distribution among the consumers.

The shared resource must be available on hosts defined in the HostExport sections.

SharedResourceExport section structure

All parameters are required.

Example SharedResourceExport section

```
Begin SharedResourceExport
NAME= AppRes
NINSTANCES= 10
DISTRIBUTION= ([C1, 30] [C2, 70])
End SharedResourceExport
```

Parameters

- NAME
- NINSTANCES
- DISTRIBUTION

NAME**Syntax**

NAME=*shared_resource_name*

Description

Shared resource to export. This resource must be available on the hosts that are exported to the specified clusters; you cannot export resources without hosts.

NINSTANCES**Syntax**

NINSTANCES=*integer*

Description

Maximum quantity of shared resource to export. If the total number available is less than the requested amount, LSF exports all that are available.

DISTRIBUTION**Syntax**

DISTRIBUTION=(*[cluster_name, number_shares]...*)

Description

Specifies how the exported resources are distributed among consumer clusters.

The syntax for the distribution list is a series of share assignments. The syntax of each share assignment is the cluster name, a comma, and the number of shares, all enclosed in square brackets, as shown. Use a space to separate multiple share assignments. Enclose the full distribution list in a set of round brackets.

cluster_name

Specify the name of a cluster allowed to use the exported resources.

number_shares

Specify a positive integer representing the number of shares of exported resources assigned to the cluster.

The number of shares assigned to a cluster is only meaningful when you compare it to the number assigned to other clusters, or to the total number. The total number of shares is the sum of all the shares assigned in each share assignment.

ResourceReservation section

By default, only LSF administrators or root can add or delete advance reservations.

The ResourceReservation section defines an advance reservation policy. It specifies:

- Users or user groups that can create reservations
- Hosts that can be used for the reservation
- Time window when reservations can be created

Each advance reservation policy is defined in a separate ResourceReservation section, so it is normal to have multiple ResourceReservation sections in `lsb.resources`.

Example ResourceReservation section

Only `user1` and `user2` can make advance reservations on `hostA` and `hostB`. The reservation time window is between 8:00 a.m. and 6:00 p.m. every day:

```
Begin ResourceReservation
NAME          = dayPolicy
USERS         = user1 user2      # optional
HOSTS         = hostA hostB     # optional
TIME_WINDOW  = 8:00-18:00      # weekly recurring reservation
End ResourceReservation
```

`user1` can add the following reservation for user `user2` to use on `hostA` every Friday between 9:00 a.m. and 11:00 a.m.:

```
% user1@hostB> brsvadd -m "hostA" -n 1 -u "user2" -t "5:9:0-5:11:0"
Reservation "user2#2" is created
```

Users can only delete reservations they created themselves. In the example, only user `user1` can delete the reservation; `user2` cannot. Administrators can delete any reservations created by users.

Parameters

- HOSTS
- NAME
- TIME_WINDOW
- USERS

HOSTS

Syntax

HOSTS=[~]host_name | [~]host_group | all | allremote | all@cluster_name ...

Description

A space-separated list of hosts, host groups defined in `lsb.hosts` on which administrators or users specified in the `USERS` parameter can create advance reservations.

The hosts can be local to the cluster or hosts leased from remote clusters.

If a group contains a subgroup, the reservation configuration applies to each member in the subgroup recursively.

Use the keyword `all` to configure reservation policies that apply to all local hosts in a cluster not explicitly excluded. This is useful if you have a large cluster but you want to use the not operator (`~`) to exclude a few hosts from the list of hosts where reservations can be created.

Use the keyword `allremote` to specify all hosts borrowed from all remote clusters.

Tip:

You cannot specify host groups or host partitions that contain the `allremote` keyword.

Use `all@cluster_name` to specify the group of all hosts borrowed from one remote cluster. You cannot specify a host group or partition that includes remote resources.

With MultiCluster resource leasing model, the not operator (`~`) can be used to exclude local hosts or host groups. You cannot use the not operator (`~`) with remote hosts.

Examples

```
HOSTS=hgroup1 ~hostA hostB hostC
```

Advance reservations can be created on `hostB`, `hostC`, and all hosts in `hgroup1` except for `hostA`.

```
HOSTS=all ~group2 ~hostA
```

Advance reservations can be created on all hosts in the cluster, except for `hostA` and the hosts in `group2`.

Default

`all allremote` (users can create reservations on all server hosts in the local cluster, and all leased hosts in a remote cluster).

NAME**Syntax**

`NAME=text`

Description

Required. Name of the `ResourceReservation` section

Specify any ASCII string 40 characters or less. You can use letters, digits, underscores (`_`) or dashes (`-`). You cannot use blank spaces.

Example

```
NAME=reservation1
```

Default

None. You must provide a name for the `ResourceReservation` section.

TIME_WINDOW

Syntax

TIME_WINDOW=*time_window* ...

Description

Optional. Time window for users to create advance reservations. The time for reservations that users create must fall within this time window.

Use the same format for *time_window* as the recurring reservation option (-t) of **brsvadd**. To specify a time window, specify two time values separated by a hyphen (-), with no space in between:

```
time_window = begin_time-end_time [time_zone]
```

Time format

Times are specified in the format:

```
[day:]hour[:minute]
```

where all fields are numbers with the following ranges:

- *day of the week*: 0-6 (0 is Sunday)
- *hour*: 0-23
- *minute*: 0-59

time_zone, which is optional, is the time zone for the time. LSF supports all standard time zone abbreviations. If you do not specify a time zone, LSF uses the local system time zone.

Specify a time window one of the following ways:

- *hour-hour* [*time_zone*]
- *hour:minute-hour:minute* [*time_zone*]
- *day:hour:minute-day:hour:minute* [*time_zone*]

The default value for minute is 0 (on the hour); the default value for day is every day of the week.

You must specify at least the hour. Day of the week and minute are optional. Both the start time and end time values must use the same syntax. If you do not specify a minute, LSF assumes the first minute of the hour (:00). If you do not specify a day, LSF assumes every day of the week. If you do specify the day, you must also specify the minute.

You can specify multiple time windows, but they cannot overlap. For example:

```
timeWindow(8:00-14:00 18:00-22:00)
```

is correct, but

```
timeWindow(8:00-14:00 11:00-15:00)
```

is not valid.

If you specify a time zone for multiple time windows, all time window entries must be consistent in whether they set the time zones. That is, either all entries must set a time zone, or all entries must not set a time zone. For example:

```
timeWindow(8:00-14:00 EDT 18:00-22:00 EDT)
```

is correct, but

```
timeWindow(8:00-14:00 18:00-22:00 EDT)
```

is not valid.

Example

```
TIME_WINDOW=8:00-14:00
```

Users can create advance reservations with begin time (**brsvadd -b**), end time (**brsvadd -e**), or time window (**brsvadd -t**) on any day between 8:00 a.m. and 2:00 p.m.

Default

Undefined (any time)

USERS**Syntax**

```
USERS=[~]user_name | [~]user_group ... | all
```

Description

A space-separated list of user names or user groups who are allowed to create advance reservations. Administrators, root, and all users or groups listed can create reservations.

If a group contains a subgroup, the reservation policy applies to each member in the subgroup recursively.

User names must be valid login names. User group names can be LSF user groups or UNIX and Windows user groups.

Use the keyword `all` to configure reservation policies that apply to all users or user groups in a cluster. This is useful if you have a large number of users but you want to exclude a few users or groups from the reservation policy.

Use the not operator (`~`) to exclude users or user groups from the list of users who can create reservations.

**CAUTION:**

The not operator does not exclude LSF administrators from the policy.

Example

```
USERS=user1 user2
```

Default

all (all users in the cluster can create reservations)

ReservationUsage section

To enable greater flexibility for reserving numeric resources that are reserved by jobs, configure the `ReservationUsage` section in `lsb.resources` to reserve resources as `PER_JOB`, `PER_TASK`, or `PER_HOST`. For example:

Example ReservationUsage section

```
Begin ReservationUsage
RESOURCE      METHOD      RESERVE
resourceX     PER_JOB    Y
resourceY     PER_HOST    N
resourceZ     PER_TASK    N
End ReservationUsage
```

Parameters

- RESOURCE
- METHOD
- RESERVE

RESOURCE

The name of the resource to be reserved. User-defined numeric resources can be reserved, but only if they are shared (they are not specific to one host).

The following built-in resources can be configured in the ReservationUsage section and reserved:

- mem
- tmp
- swp

Any custom resource can also be reserved if it is shared (defined in the Resource section of `lsf.shared`) or host based (listed in the Host section of the `lsf.cluster` file in the resource column).

METHOD

The resource reservation method. One of:

- PER_JOB
- PER_HOST
- PER_TASK

The cluster-wide `RESOURCE_RESERVE_PER_SLOT` parameter in `lsb.params` is obsolete.

`RESOURCE_RESERVE_PER_TASK` parameter still controls resources not configured in `lsb.resources`. Resources not reserved in `lsb.resources` are reserved by job if it is a shared resource or by host if it is a host-based resource.

`PER_HOST` reservation means that for the parallel job, LSF reserves one instance of a for each host. For example, some application licenses are charged only once no matter how many applications are running provided those applications are running on the same host under the same user.

Use no method ("-") when setting mem, swp, or tmp as `RESERVE=Y`.

RESERVE

Reserves the resource for pending jobs that are waiting for another resource to become available.

For example, job A requires resources X, Y, and Z to run, but resource Z is a high demand or scarce resource. This job pends until Z is available. In the meantime, other jobs requiring only X and Y resources run. If X and Y are set as reservable resources (the **RESERVE** parameter is set to "Y"), as soon as Z resource is available, job A runs. If they are not, job A may never be able to run because all resources are never available at the same time.

Restriction:

Only the following built-in resources can be defined as reservable:

- mem
- swp
- tmp

Use no method (-) when setting mem, swp, or tmp as `RESERVE=Y`.

When submitting a job, the queue must have `RESOURCE_RESERVE` defined.

Backfill of the reservable resources is also supported when you submit a job with reservable resources to a queue with `BACKFILL` defined.

Valid values are Y and N. If not specified, resources are not reserved.

Assumptions and limitations

- Per-resource configuration defines resource usage for individual resources, but it does not change any existing resource limit behavior (PER_JOB, PER_TASK).
- In the LSF multicluster capability environment, you should configure resource usage in the scheduling cluster (submission cluster in lease model or receiving cluster in job forward model).

Automatic time-based configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in `lsb.resources` by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the **admin reconfig** command.

The expressions are evaluated by LSF every 10 minutes based on `mbatchd` start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting `mbatchd`, providing continuous system availability.

Example

```
# limit usage of hosts for group and time
# based configuration
# - 10 jobs can run from normal queue
# - any number can run from short queue between 18:30
#   and 19:30
#   all other hours you are limited to 100 slots in the
#   short queue
# - each other queue can run 30 jobs
```

```
Begin Limit
PER_QUEUE          HOSTS          SLOTS      # Example
normal             Resource1    10
# if time(18:30-19:30 EDT)
short              Resource1    -
#else
short              Resource1    100
#endif
(all ~normal ~short) Resource1    30
End Limit
```

Specifying the time zone is optional. If you do not specify a time zone, LSF uses the local system time zone. LSF supports all standard time zone abbreviations.

PowerPolicy section

This section enables and defines a power management policy.

Example PowerPolicy section

```
Begin PowerPolicy
NAME              = policy_night
HOSTS              = hostGroup1 host3
TIME_WINDOW       = 23:00-8:00 EDT
MIN_IDLE_TIME     = 1800
CYCLE_TIME        = 60
End PowerPolicy
```

Parameters

- NAME
- HOSTS
- TIME_WINDOW

- MIN_IDLE_TIME
- CYCLE_TIME

NAME

Syntax

NAME=*string*

Description

Required. Unique name for the power management policy.

Specify any ASCII string 60 characters or less. You can use letters, digits, underscores (_), dashes (-), or periods (.). You cannot use blank spaces.

Example

```
NAME=policy_night1
```

Default

None. You must provide a name to define a power policy.

HOSTS

Syntax

HOSTS=*host_list*

Description

host_list is a space-separated list of host names, host groups, host partitions, or compute units.

Required. Specified hosts should not overlap between power policies.

Example

```
HOSTS=hostGroup1 host3
```

Default

If a host is not defined, the default value is all other hosts that are not included in power policy. (Does not contain the master and master candidate)

TIME_WINDOW

Syntax

TIME_WINDOW=*time_window* ...

Description

Required. Time window is the time period to which the power policy applies.

To specify a time window, specify two time values separated by a hyphen (-), with no space in between

```
time_window = begin_time-end_time
```

Time format

Times are specified in the format:

```
[day:]hour[:minute]
```

where all fields are numbers with the following ranges:

- *day of the week*: 0-6 (0 is Sunday)
- *hour*: 0-23
- *minute*: 0-59

Specify a time window one of the following ways:

- *hour-hour*
- *hour:minute-hour:minute*
- *day:hour:minute-day:hour:minute*

The default value for minute is 0 (on the hour); the default value for day is every day of the week.

You must specify at least the hour. Day of the week and minute are optional. Both the start time and end time values must use the same syntax. If you do not specify a minute, LSF assumes the first minute of the hour (:00). If you do not specify a day, LSF assumes every day of the week. If you do specify the day, you must also specify the minute.

You can specify multiple time windows, but they cannot overlap. For example:

```
timeWindow(8:00-14:00 18:00-22:00)
```

is correct, but

```
timeWindow(8:00-14:00 11:00-15:00)
```

is not valid.

Example

```
TIME_WINDOW=8:00-14:00
```

Default

Not defined (any time)

MIN_IDLE_TIME**Syntax**

```
MIN_IDLE_TIME=minutes
```

Description

This parameter takes effect if **TIME_WINDOW** is configured and is valid. It defines the host idle time before power operations are issued for defined hosts.

Example

```
MIN_IDLE_TIME=60
```

Default

0

CYCLE_TIME

Syntax

CYCLE_TIME=minutes

Description

This parameter takes effect if **TIME_WINDOW** is configured and is valid. It defines the minimum time (in minutes) between changes in power states for defined hosts.

Example

```
CYCLE_TIME=15
```

Default

0

lsb.serviceclasses

The `lsb.serviceclasses` file defines the service-level agreements (SLAs) in an LSF cluster as *service classes*, which define the properties of the SLA.

This file is optional.

You can configure as many service class sections as you need.

Use **bsla** to display the properties of service classes configured in `lsb.serviceclasses` and dynamic information about the state of each configured service class.

By default, `lsb.serviceclasses` is installed in `LSB_CONFDIR/cluster_name/configdir`.

Changing lsb.serviceclasses configuration

After making any changes to `lsb.serviceclasses`, run **badm** **reconfig** to reconfigure `mbatchd`.

lsb.serviceclasses structure

Each service class definition begins with the line `Begin ServiceClass` and ends with the line `End ServiceClass`.

Syntax

```

Begin ServiceClass
NAME           = string
PRIORITY       = integer
GOALS          = [throughput | velocity | deadline] [\...]
CONTROL_ACTION = VIOLATION_PERIOD[minutes] CMD [action]
USER_GROUP     = all | [user_name] [user_group] ...
DESCRIPTION    = text

End ServiceClass

Begin ServiceClass

NAME           = string
GOALS          = guarantee
ACCESS_CONTROL = [QUEUES[ queue ...]] [USERS[ [user_name] [user_group] ...]]
               [FAIRSHARE_GROUPS[user_group ...]] [APPS[app_name ...]]
               [PROJECTS[proj_name...]]
AUTO_ATTACH   = Y | y | N | n
DESCRIPTION    = text

```

```
End ServiceClass
```

You must specify:

- Service class name
- Goals

Service classes with guarantee goals cannot have **PRIORITY**, **CONTROL_ACTION** or **USER_GROUP** defined.

To configure EGO-enabled SLA scheduling, you must specify an existing EGO consumer name to allow the SLA to get host allocations from EGO.

All other parameters are optional.

Example

```
Begin ServiceClass
NAME=Sooke
PRIORITY=20
GOALS=[DEADLINE timeWindow (8:30-16:00)]
DESCRIPTION="working hours"
End ServiceClass

Begin ServiceClass
NAME=Newmarket
GOALS=[GUARANTEE]
ACCESS_CONTROL = QUEUES[batch] FAIRSHARE_GROUPS[team2]
AUTO_ATTACH = Y
DESCRIPTION="guarantee for team2 batch jobs"
End ServiceClass
```

Parameters

- ACCESS_CONTROL
- AUTO_ATTACH
- CONSUMER
- CONTROL_ACTION
- DESCRIPTION
- EGO_RES_REQ
- EGO_RESOURCE_GROUP
- GOALS
- MAX_HOST_IDLE_TIME
- NAME
- PRIORITY
- USER_GROUP

ACCESS_CONTROL

Syntax

```
ACCESS_CONTROL=[QUEUES[queue ...]] [USERS[ [user_name] [user_group] ...]]
[FAIRSHARE_GROUPS[user_group ...]] [APPS[app_name ...]] [PROJECTS[proj_name...]]
[LIC_PROJECTS[lic_proj...]]
```

Description

Guarantee SLAs (with GOALS=[GUARANTEE]) only.

Restricts access to a guarantee SLA. If more than one restriction is configured, all must be satisfied.

- **QUEUES** restricts access to the queues listed; the queue is specified for jobs at submission using **bsub -q**.
- **USERS** restricts access to jobs submitted by the users or user groups specified.

User names must be valid login names. To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*). User group names can be LSF user groups or UNIX and Windows user groups. To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_group*).

- **FAIRSHARE_GROUPS** restricts access to the fairshare groups listed; the fairshare group is specified for jobs at submission using **bsub -G**.
- **APPS** restricts access to the application profiles listed; the application profile is specified for jobs at submission using **bsub -app**.
- **PROJECTS** restricts access to the projects listed; the project is specified for jobs at submission using **bsub -P**.

Example

```
ACCESS_CONTROL = QUEUES[normal short] USERS[ug1]
```

Jobs submitted to the queues normal or short by users in usergroup ug1 are the only jobs accepted by the guarantee SLA.

Default

None. Access to the guarantee SLA is not restricted.

AUTO_ATTACH

Syntax

AUTO_ATTACH=Y | y | N | n

Description

Guarantee SLAs (with GOALS=[GUARANTEE]) only. Used with **ACCESS_CONTROL**.

Enabling **AUTO_ATTACH** when a guarantee SLA has **ACCESS_CONTROL** configured results in submitted jobs automatically attaching to the guarantee SLA if they have access. If a job can access multiple guarantee SLAs with **AUTO_ATTACH** enabled, the job is automatically attached to the first accessible SLA based on configuration order in the `lsb.serviceclasses` file.

During restart or reconfiguration, automatic attachments to guarantee SLAs are checked and jobs may be attached to a different SLA. During live reconfiguration (using the **bconf** command) automatic attachments are not checked, and jobs remain attached to the same guarantee SLAs regardless of configuration changes.

Example

```
Begin ServiceClass
...
NAME = Maple
GOALS = [GUARANTEE]
ACCESS_CONTROL = QUEUES[priority] USERS[ug1]
AUTO_ATTACH = Y
```

```
...  
End ServiceClass
```

All jobs submitted to the priority queue by users in user group ug1 and submitted without an SLA specified are automatically attached to the service class Maple.

Default

N

CONSUMER

Syntax

CONSUMER=*ego_consumer_name*

Description

For EGO-enabled SLA service classes, the name of the EGO consumer from which hosts are allocated to the SLA. This parameter is not mandatory, but must be configured for the SLA to receive hosts from EGO.

Guarantee SLAs (with GOALS=[GUARANTEE]) cannot have **CONSUMER** set. If defined, it will be ignored.

Important: CONSUMER must specify the name of a valid consumer in EGO. If a default SLA is configured with ENABLE_DEFAULT_EGO_SLA in lsb.params, all services classes configured in lsb.serviceclasses must specify a consumer name.

Default

None

CONTROL_ACTION

Syntax

CONTROL_ACTION=**VIOLATION_PERIOD** [*minutes*] **CMD** [*action*]

Description

Optional. Configures a control action to be run if the SLA goal is delayed for a specified number of minutes.

If the SLA goal is delayed for longer than VIOLATION_PERIOD, the action specified by CMD is invoked. The violation period is reset and if the SLA is still active when the violation period expires again, the action runs again. If the SLA has multiple active goals that are in violation, the action is run for each of them.

Guarantee SLAs (with GOALS=[GUARANTEE]) cannot have **CONTROL_ACTION** set. If defined, it will be ignored.

Example

```
CONTROL_ACTION=VIOLATION_PERIOD[10] CMD [echo `date`: SLA is in violation >>  
! /tmp/sla_violation.log]
```

Default

None

DESCRIPTION

Syntax

DESCRIPTION=*text*

Description

Optional. Description of the service class. Use **bsla** to display the description text.

This description should clearly describe the features of the service class to help users select the proper service class for their jobs.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\).

Default

None

EGO_RES_REQ

Syntax

EGO_RES_REQ=*res_req*

Description

For EGO-enabled SLA service classes, the EGO resource requirement that specifies the characteristics of the hosts that EGO will assign to the SLA.

Must be a valid EGO resource requirement. The EGO resource requirement string supports the select section, but the format is different from LSF resource requirements.

Guarantee SLAs (with GOALS=[GUARANTEE]) cannot have **EGO_RES_REQ** set. If defined, it will be ignored.

Note: After changing this parameter, running jobs using the allocation may be re-queued.

Example

```
EGO_RES_REQ=select(linux && maxmem > 100)
```

Default

None

EGO_RESOURCE_GROUP

Syntax

EGO_RESOURCE_GROUP=*ego resource group name* or a blank separated group list

Description

For EGO-enabled SLA service classes. A resource group or space-separated list of resource groups from which hosts are allocated to the SLA.

List must be a subset of or equal to the resource groups allocated to the consumer defined by the **CONSUMER** entry.

Guarantee SLAs (with GOALS=[GUARANTEE]) cannot have **EGO_RESOURCE_GROUP** set. If defined, it will be ignored.

Note: After changing this parameter, running jobs using the allocation may be re-queued.

Example

```
EGO_RESOURCE_GROUP=resource_group1 resource_group4 resource_group5
```

Default

Undefined (**venkd** determines which resource groups to allocate slots to LSF).

GOALS

Syntax

```
GOALS=[throughput | velocity | deadline] [\
throughput | velocity | deadline] ...]
GOALS=[guarantee]
```

Description

Required. Defines the service-level goals for the service class. A service class can have more than one goal, each active at different times of the day and days of the week. Outside of the time window, the SLA is inactive and jobs are scheduled as if no service class is defined. LSF does not enforce any service-level goal for an inactive SLA.

The time windows of multiple service-level goals can overlap. In this case, the largest number of jobs is run.

An active SLA can have a status of On time if it is meeting the goal, and a status Delayed, if it is missing its goals.

A service-level goal defines:

throughput - expressed as *finished* jobs per hour and an optional time window when the goal is active. *throughput* has the form:

```
GOALS=[THROUGHPUT num_jobs timeWindow [(time_window)]]
```

If no time window is configured, THROUGHPUT can be the only goal in the service class. The service class is always active, and **bsla** displays ACTIVE WINDOW: Always Open.

velocity - expressed as *concurrently* running jobs and an optional time window when the goal is active. *velocity* has the form:

```
GOALS=[VELOCITY num_jobs timeWindow [(time_window)]]
```

If no time window is configured, VELOCITY can be the only goal in the service class. The service class is always active, and **bsla** displays ACTIVE WINDOW: Always Open.

deadline - indicates that all jobs in the service class should complete by the end of the specified time window. The time window is required for a deadline goal. *deadline* has the form:

```
GOALS=[DEADLINE timeWindow (time_window)]
```

guarantee - indicates the SLA has guaranteed resources defined in `lsb.resources` and is able to guarantee resources, depending on the scavenging policies configured. Guarantee goals cannot be combined with any other goals, and do not accept time windows.

```
GOALS=[GUARANTEE]
```

Restriction: EGO-enabled SLA service classes only support velocity goals. Deadline, throughput, and guarantee goals are not supported. The configured velocity value for EGO-enabled SLA service classes is considered to be a *minimum* number of jobs that should be in run state from the SLA

Time window format

The time window of an SLA goal has the standard form:

```
begin_time-end_time
```

Times are specified in the format:

```
[day:]hour[:minute]
```

where all fields are numbers with the following ranges:

- *day of the week*: 0-6 (0 is Sunday)
- *hour*: 0-23
- *minute*: 0-59

Specify a time window one of the following ways:

- *hour-hour*
- *hour:minute-hour:minute*
- *day:hour:minute-day:hour:minute*

The default value for minute is 0 (on the hour); the default value for day is every day of the week.

You must specify at least the hour. Day of the week and minute are optional. Both the start time and end time values must use the same syntax. If you do not specify a minute, LSF assumes the first minute of the hour (:00). If you do not specify a day, LSF assumes every day of the week. If you do specify the day, you must also specify the minute.

You can specify multiple time windows, but they cannot overlap. For example:

```
timeWindow(8:00-14:00 18:00-22:00)
```

is correct, but

```
timeWindow(8:00-14:00 11:00-15:00)
```

is not valid.

Tip:

To configure a time window that is always open, use the `timeWindow` keyword with empty parentheses.

Examples

```
GOALS=[THROUGHPUT 2 timeWindow ()]
```

```
GOALS=[THROUGHPUT 10 timeWindow (8:30-16:30)]
```

```
GOALS=[VELOCITY 5 timeWindow ()]
```

```
GOALS=[DEADLINE timeWindow (16:30-8:30)] [VELOCITY 10 timeWindow (8:30-16:30)]
```

```
GOALS=[GUARANTEE]
```

MAX_HOST_IDLE_TIME

Syntax

```
MAX_HOST_IDLE_TIME=seconds
```

Description

For EGO-enabled SLA service classes, number of seconds that the SLA will hold its idle hosts before LSF releases them to EGO. Each SLA can configure a different idle time. Do not set this parameter to a small value, or LSF may release hosts too quickly.

Guarantee SLAs (with GOALS=[GUARANTEE]) cannot have **MAX_HOST_IDLE_TIME** set. If defined, it will be ignored.

Default

120 seconds

NAME

Syntax

```
NAME=string
```

Description

Required. A unique name that identifies the service class.

Specify any ASCII string 60 characters or less. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces.

Important:

The name you use cannot be the same as an existing host partition, user group name, or fairshare queue name.

Example

```
NAME=Tofino
```

Default

None. You must provide a unique name for the service class.

PRIORITY

Syntax

```
PRIORITY=integer
```

Description

Required (time-based SLAs only). The service class priority. A higher value indicates a higher priority, relative to other service classes. Similar to queue priority, service classes access the cluster resources in priority order.

LSF schedules jobs from one service class at a time, starting with the highest-priority service class. If multiple service classes have the same priority, LSF runs all the jobs from these service classes in first-come, first-served order.

Service class priority in LSF is completely independent of the UNIX scheduler's priority system for time-sharing processes. In LSF, the NICE parameter is used to set the UNIX time-sharing priority for batch jobs.

Guarantee SLAs (with GOALS=[GUARANTEE]) cannot have **PRIORITY** set. If defined, it will be ignored.

Default

None.

USER_GROUP

Syntax

```
USER_GROUP=all | [user_name] [user_group] ...
```

Description

Optional. A space-separated list of user names or user groups who can submit jobs to the service class. Administrators, root, and all users or groups listed can use the service class.

Use the reserved word all to specify all LSF users. LSF cluster administrators are automatically included in the list of users, so LSF cluster administrators can submit jobs to any service class, or switch any user's jobs into this service class, even if they are not listed.

If user groups are specified in `lsb.users`, each user in the group can submit jobs to this service class. If a group contains a subgroup, the service class policy applies to each member in the subgroup recursively. If the group can define fairshare among its members, the SLA defined by the service class enforces the fairshare policy among the users of the SLA.

User names must be valid login names. User group names can be LSF user groups (in `lsb.users`) or UNIX and Windows user groups.

Guarantee SLAs (with GOALS=[GUARANTEE]) cannot have **USER_GROUP** set. If defined, it will be ignored.

Example

```
USER_GROUP=user1 user2 ugroup1
```

Default

all (all users in the cluster can submit jobs to the service class)

Examples

- The resource-based service class AccountingSLA guarantees hosts to the user group accountingUG for jobs submitted to the queue longjobs. Jobs submitted to this queue by this usergroup without an SLA specified will be automatically attached to the SLA. The guaranteed resource pools used by the SLA are configured in `lsb.resources`.

```
Begin ServiceClass
NAME=AccountingSLA
GOALS=[GUARANTEE]
DESCRIPTION="Guaranteed hosts for the accounting department"
ACCESS_CONTROL = QUEUES[longjobs] USERS[accountingUG]
AUTO_ATTACH = Y
End ServiceClass
```

- The service class Sooke defines one deadline goal that is active during working hours between 8:30 AM and 4:00 PM. All jobs in the service class should complete by the end of the specified time window. Outside of this time window, the SLA is inactive and jobs are scheduled without any goal being enforced:

```
Begin ServiceClass
NAME=Sooke
PRIORITY=20
GOALS=[DEADLINE timeWindow (8:30-16:00)]
DESCRIPTION="working hours"
End ServiceClass
```

- The service class Nanaimo defines a deadline goal that is active during the weekends and at nights.

```
Begin ServiceClass
NAME=Nanaimo
PRIORITY=20
GOALS=[DEADLINE timeWindow (5:18:00-1:8:30 20:00-8:30)]
DESCRIPTION="weekend nighttime regression tests"
End ServiceClass
```

- The service class Sidney defines a throughput goal of 6 jobs per hour that is always active:

```
Begin ServiceClass
NAME=Sidney
PRIORITY=20
GOALS=[THROUGHPUT 6 timeWindow ()]
DESCRIPTION="constant throughput"
End ServiceClass
```

- The service class Tofino defines two velocity goals in a 24 hour period. The first goal is to have a maximum of 10 concurrently running jobs during business hours (9:00 a.m. to 5:00 p.m). The second goal is a maximum of 30 concurrently running jobs during off-hours (5:30 p.m. to 8:30 a.m.)

```
Begin ServiceClass
NAME=Tofino
PRIORITY=20
GOALS=[VELOCITY 10 timeWindow (9:00-17:00)] [VELOCITY 30 timeWindow (17:30-8:30)]
DESCRIPTION="day and night velocity"
End ServiceClass
```

- The service class Duncan defines a velocity goal that is active during working hours (9:00 a.m. to 5:30 p.m.) and a deadline goal that is active during off-hours (5:30 p.m. to 9:00 a.m.) Only users user1 and user2 can submit jobs to this service class.

```
Begin ServiceClass
NAME=Duncan
PRIORITY=23
USER_GROUP=user1 user2
GOALS=[VELOCITY 8 timeWindow (9:00-17:30)] [DEADLINE timeWindow (17:30-9:00)]
DESCRIPTION="Daytime/Nighttime SLA"
End ServiceClass
```

- The service class `Tevere` defines a combination similar to `Duncan`, but with a deadline goal that takes effect overnight and on weekends. During the working hours in weekdays the velocity goal favors a mix of short and medium jobs.

```
Begin ServiceClass
NAME=Tevere
PRIORITY=20
GOALS=[VELOCITY 100 timeWindow (9:00-17:00)] [DEADLINE timeWindow (17:30-8:30
5:17:30-1:8:30)]
DESCRIPTION="nine to five"
End ServiceClass
```

lsb.threshold

The `lsb.threshold` configuration file defines energy-saving and CPU frequency policies. This file is optional.

The threshold file (`lsb.threshold`) is available at the location specified by the parameter **PERFORMANCE_THRESHOLD_FILE** in the `lsb.params` file. The default location is `$LSF_ENVDIR/lsbatch/cluster_name/configdir/lsb.threshold`.

The threshold file has the following parts:

Minimize energy policy

The purpose of this policy is to save energy. With this policy, LSF will select a frequency that is equal to or less than the default CPU frequency and the time variation is equal to or less than the threshold value, which can save the most energy. The value should be a positive value or 0. The default threshold value is 0.

Minimize time policy

The purpose of this policy is to improve performance. The value must be a negative value. This policy allows the job to run at a frequency that is higher than the default frequency. The policy is only available when the default frequency is less than the nominal frequency.

LSF only considers the frequency range that is higher than the default frequency. When a job runs with this policy, LSF checks the time variation from the nominal to the default frequency. The frequency is used for the job when the time variation is less than or equal to the corresponding threshold value (the absolute value of time variation is large than or equal to corresponding threshold's absolute value); otherwise, LSF checks the next frequency.

The job will run in the default frequency when there is no frequency that matches the specified threshold value. You must specify at least one frequency value that is larger than the default CPU frequency and less than or equal to the maximum available frequency. The available frequency list is available at `/sys/devices/system/cpu/cpu*/cpufreq/scaling_available_frequencies`. If it is not in the available frequency list, the frequency will be ignored. For other frequencies larger than the default CPU frequency, LSF automatically calculates the value based on the difference of default frequency and defined frequency. LSF will calculate the increase between each frequency and use that to get all frequencies that are larger than the default CPU frequency.

The following is an example `lsb.threshold` file:

```
# Copyright International Business Machines Corp,1993-2006, 2013
#....
# Minimize energy policy
#....
Begin Min_Energy
THRESHOLD_RUNTIME_VAR = 10
End Min_Energy
#
# Minimize run time policy
#...
Begin Min_Time
CPU_FREQ      RUNTIME_VAR
```

lsb.users

```
2100000KHz -2
2200000KHz -4
2300000KHz -9
2400000KHz -12
2500000KHz -15
2700000KHz -20
End Min_Time
```

Example definitions are commented with a pound sign (#) in the sample `lsb.threshold` file. Remove the # to enter your own values for the example definitions. If no lines are uncommented, default values will be used for all definitions.

lsb.users

The `lsb.users` file is used to configure user groups, hierarchical fairshare for users and user groups, and job slot limits for users and user groups. It is also used to configure account mappings in a MultiCluster environment.

This file is optional.

The `lsb.users` file is stored in the directory `LSB_CONFDIR/cluster_name/configdir`, where `LSB_CONFDIR` is defined in `lsf.conf`.

Changing lsb.users configuration

After making any changes to `lsb.users`, run **badadmin reconfig** to reconfigure mbatchd.

#INCLUDE

Syntax

```
#INCLUDE "path-to-file"
```

Description

Inserts a configuration setting from another file to the current location. Use this directive to dedicate control of a portion of the configuration to other users or user groups by providing write access for the included file to specific users or user groups, and to ensure consistency of configuration file settings in different clusters (if you are using the LSF multicluster capability).

See more information on shared configuration file content in *Administering IBM Spectrum LSF*

#INCLUDE can be inserted anywhere in the local configuration file.

Default

Not defined.

UserGroup section

Optional. Defines user groups.

The name of the user group can be used in other user group and queue definitions, as well as on the command line. Specifying the name of a user group in the `GROUP_MEMBER` section has exactly the same effect as listing the names of all users in the group.

The total number of user groups cannot be more than 1024.

Structure

The first line consists of two mandatory keywords, `GROUP_NAME` and `GROUP_MEMBER`. Other keywords are optional. Subsequent lines name a group and list its membership and optionally its share assignments and administrator.

Each line must contain one entry for each keyword. Use empty parentheses () or a dash - to specify the default value for an entry.

Restriction: If specifying a specific user name for a user group, that entry must precede all user groups.

Examples of a UserGroup section

Example 1:

```
Begin UserGroup
GROUP_NAME      GROUP_MEMBER      GROUP_ADMIN      PRIORITY
groupA          (user1 user2 user3 user4) (user5[full])    100
groupB          (user7 user8 user9)    (groupA[usershares]) 50
groupC          (groupA user5)          (groupA)          50
groupD          (!)                     ( )                10
```

Example 2:

```
Begin UserGroup
GROUP_NAME      GROUP_MEMBER      GROUP_ADMIN      PRIORITY
groupA          (user1 user2 user3 user4) (user5)          200
groupB          (groupA user5)      (groupA)          100
groupC          (!)                 ( )                50
End UserGroup
```

Example 3:

```
Begin UserGroup
GROUP_NAME      GROUP_MEMBER      USER_SHARES      PRIORITY
groupB          (user1 use r2)    ( )                100
groupC          (user3 user4)      ([User3,3] [User4,4]) 200
groupA          (GroupB GroupC user5) ([User5,1] [default,10]) 50
End UserGroup
```

GROUP_NAME

An alphanumeric string representing the user group name. You cannot use the reserved name all or a "/" in a group name.

FS_POLICY

Syntax

(global_fairshare_policy_name ...)

Description

Optional. Enables partial **SYNC_MODE** global fairshare policy for the defined user group. It specifies which partial **SYNC_MODE** global fairshare policy the share account will participate into.

Default

Not defined.

Example

```
Begin UserGroup
GROUP_NAME      GROUP_MEMBER      USER_SHARES      FS_POLICY
group1          (user1 user2)    ([default, 10])    (utopia)
End UserGroup
```

GROUP_MEMBER

User group members are the users who belong to the group. You can specify both user names and user group names.

User and user group names can appear on multiple lines because users can belong to multiple groups.

Note:

When a user belongs to more than one group, any of the administrators specified for any of the groups the user belongs to can control that users' jobs. Limit administrative control by defining `STRICT_UG_CONTROL=Y` in `lsb.params` and submitting jobs with the `-G` option, specifying which user group the job is submitted with.

User groups may be defined recursively but must not create a loop.

Syntax

(user_name | user_group ...) | (all) | (!)

Enclose the entire group member list in parentheses. Use space to separate multiple names.

You can combine user names and user group names in the same list.

Valid values

- all

The reserved name all specifies all users in the cluster.

- !

An exclamation mark (!) indicates an externally-defined user group, which the **egroup** executable retrieves.

- *user_name*

User names must be valid login names.

To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME \user_name*).

- *user_group*

User group names can be LSF user groups defined previously in this section, or UNIX and Windows user groups.

If you specify a name that is both a UNIX user group and also a UNIX user, append a backslash to make sure it is interpreted as a group (*user_group/*).

To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME \user_group*).

GROUP_ADMIN

User group administrators can administer the jobs of group members. You can specify both user names and user group names.

- If you specify a user group as an administrator for another user group, all members of the first user group become administrators for the second user group.
- You can also specify that all users of a group are also administrators of that same group.
- Users can be administrators for more than one user group at the same time.

Note:

When a user belongs to more than one group, any of the administrators specified for any of the groups the user belongs to can control that users' jobs. Define `STRICT_UG_CONTROL=Y` in `lsb.params` to limit user group administrator control to the user group specified by `-G` at job submission.

By default a user group administrator has privileges equivalent to those of a job owner, and is allowed to control any job belonging to member users of the group they administer. A user group administrator can also resume jobs stopped by the LSF administrator or queue administrator if the job belongs to a member of their user group.

Optionally, you can specify additional user group administrator rights for each user group administrator.

User group administrator rights are inherited. For example, if admin2 has full rights for user group ugA and user group ugB is a member of ugA, admin2 also has full rights for user group ugB.

Restriction:

Unlike a job owner, a user group administrator cannot run **brestart** and **bread -a data_file**.

To manage security concerns, you cannot specify user group administrators for any user group containing the keyword all as a member unless STRICT_UG_CONTROL=Y is defined in lsb.params.

Syntax

(*user_name* | *user_name*[*admin_rights*] | *user_group* | *user_group*[*admin_rights*] ...)

Enclose the entire group administrator list in parentheses. If you specify administrator rights for a user or group, enclose them in square brackets.

You can combine user names and user group names in the same list. Use space to separate multiple names.

Valid values

- *user_name*

User names must be valid login names.

To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME \user_name*).

- *user_group*

User group names can be LSF user groups defined previously in this section, or UNIX and Windows user groups.

If you specify a name that is both a UNIX user group and also a UNIX user, append a backslash to make sure it is interpreted as a group (*user_group/*).

To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME \user_group*).

- *admin_rights*

- If no rights are specified, only default job control rights are given to user group administrators.

- usershares: user group administrators with usershares rights can adjust user shares using **bconf update**.

- full: user group administrators with full rights can use **bconf** to delete the user group, create new user groups, and adjust usershares, group members, and user group priorities.

User group administrators with full rights can only add a user group member to the user group if they also have full rights for the member user group.

User group administrators adding a new user group with **bconf create** are automatically added to GROUP_ADMIN with full rights for the new user group.

Restrictions

- Wildcard and special characters are not supported (for example: *, !, \$, #, &, ~)
- The reserved keywords others, default, allremote are not supported.
- User groups with the keyword all as a member can only have user group administrators configured if STRICT_UG_CONTROL=Y is defined in lsb.params.
- User groups with the keyword all as a member cannot be user group administrators.
- User groups and user groups administrator definitions cannot be recursive or create a loop.

PRIORITY

Optional. Specifies a priority that is used as a factor when calculating the job priority for absolute job priority scheduling (APS).

Valid values

Specify an integer between 0 and 2147483646.

Default

Not defined.

If APS is enabled for users, user groups, or application profiles, the default value is 0.

USER_SHARES

Optional. Enables hierarchical fairshare and defines a share tree for users and user groups.

By default, when resources are assigned collectively to a group, the group members compete for the resources according to FCFS scheduling. You can use hierarchical fairshare to further divide the shares among the group members.

Syntax

([*user*, *number_shares*])

Specify the arguments as follows:

- Enclose the list in parentheses, even if you do not specify any user share assignments.
- Enclose each user share assignment in square brackets, as shown.
- Separate the list of share assignments with a space.
- *user*—Specify users or user groups. You can assign the shares to:
 - A single user (specify *user_name*). To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).
 - Users in a group (specify *group_name*). To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME\group_name*).
 - Users not included in any other share assignment, individually (specify the keyword *default*) or collectively (specify the keyword *others*).

Note: By default, when resources are assigned collectively to a group, the group members compete for the resources on a first-come, first-served (FCFS) basis. You can use hierarchical fairshare to further divide the shares among the group members. When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.

- *number_shares*—Specify a positive integer representing the number of shares of the cluster resources assigned to the user. The number of shares assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares assigned in each share assignment.

User section

Optional. If this section is not defined, all users and user groups can run an unlimited number of jobs in the cluster.

This section defines the maximum number of jobs a user or user group can run concurrently in the cluster. This is to avoid situations in which a user occupies all or most of the system resources while other users' jobs are waiting.

Structure

One field is mandatory: USER_NAME.

All other fields are optional.

You must specify a dash (-) to indicate the default value (unlimited) if a user or user group is specified. Fields cannot be left blank.

Example of a User section

```

Begin User
USER_NAME  MAX_JOBS  JL/P  MAX_PEND_JOBS  PRIORITY
user1      10        -      1000          200
user2      4         -        -             100
user3      -         -        -             300
groupA     10        1      100000        400
groupA@    -         1       100           200
groupC     -         -        500           100
default    6         1       10             0
End User

```

USER_NAME

User or user group for which job slot limits are defined.

Use the reserved user name default to specify a job slot limit that applies to each user and user group not explicitly named. Since the limit specified with the keyword default applies to user groups also, make sure you select a limit that is high enough, or explicitly define limits for user groups.

User group names can be the LSF user groups defined previously, and/or UNIX and Windows user groups. To specify a Windows user account or user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_name* or *DOMAIN_NAME\user_group*).

Job slot limits apply to a group as a whole. Append the at sign (@) to a group name to make the job slot limits apply individually to each user in the group. If a group contains a subgroup, the job slot limit also applies to each member in the subgroup recursively.

If the group contains the keyword all in the user list, the at sign (@) has no effect. To specify job slot limits for each user in a user group containing all, use the keyword default.

MAX_JOBS

Per-user or per-group job slot limit for the cluster. Total number of job slots that each user or user group can use in the cluster.

Note: If a group contains the keyword all as a member, all users and user groups are included in the group. The per-group job slot limit set for the group applies to the group as a whole, limiting the entire cluster even when **ENFORCE_ONE_UG_LIMIT** is set in `lsb.params`.

JL/P

Per processor job slot limit per user or user group.

Total number of job slots that each user or user group can use per processor. This job slot limit is configured per processor so that multiprocessor hosts will automatically run more jobs.

This number can be a fraction such as 0.5, so that it can also serve as a per-host limit. This number is rounded up to the nearest integer equal to or greater than the total job slot limits for a host. For example, if JL/P is 0.5, on a 4-CPU multiprocessor host, the user can only use up to 2 job slots at any time. On a uniprocessor machine, the user can use 1 job slot.

MAX_PEND_JOBS

Per-user or per-group pending job limit. This is the total number of pending jobs that each user or user group can have in the system. If a user is a member of multiple user groups, the user's pending jobs are counted towards the pending job limits of all groups from which the user has membership.

If **ENFORCE_ONE_UG_LIMITS** is set to Y in `lsb.params` and you submit a job while specifying a user group, only the limits for that user group (or any parent user group) apply to the job even if there are overlapping user group members.

MAX_PEND_SLOTS

Per-user or per-group pending job slot limit. This is the total number of pending job slots that each user or user group can have in the system. If a user is a member of multiple user groups, the user's pending jobs are counted towards the pending job slot limits of all groups from which the user has membership.

If **ENFORCE_ONE_UG_LIMITS** is set to Y in `lsb.params` and you submit a job while specifying a user group, only the limits for that user group (or any parent user group) apply to the job even if there are overlapping user group members.

PRIORITY

Optional. Specifies a priority that is used as a factor when calculating the job priority for absolute job priority scheduling (APS).

Valid values

Specify an integer between 0 and 2147483646.

Default

Not defined.

If APS is enabled for users, user groups, or application profiles, the default value is 0.

UserMap section

Optional. Used only in a MultiCluster environment with a non-uniform user name space. Defines system-level cross-cluster account mapping for users and user groups, which allows users to submit a job from a local host and run the job as a different user on a remote host. Both the local and remote clusters must have corresponding user account mappings configured.

Structure

The following three fields are all required:

- LOCAL
- REMOTE
- DIRECTION

LOCAL

A list of users or user groups in the local cluster. To specify a Windows user account or user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_name* or *DOMAIN_NAME\user_group*). Separate multiple user names by a space and enclose the list in parentheses ():

```
(user4 user6)
```

REMOTE

A list of remote users or user groups in the form *user_name@cluster_name* or *user_group@cluster_name*. To specify a Windows user account or user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_name@cluster_name* or *DOMAIN_NAME\user_group@cluster_name*). Separate multiple user names by a space and enclose the list in parentheses ():

```
(user4@cluster2 user6@cluster2)
```

DIRECTION

Specifies whether the user account runs jobs locally or remotely. Both directions must be configured on the local and remote clusters.

- The export keyword configures local users/groups to run jobs as remote users/groups.
- The import keyword configures remote users/groups to run jobs as local users/groups.

Example of a UserMap section

On cluster1:

```
Begin UserMap
LOCAL    REMOTE                DIRECTION
user1    user2@cluster2        export
user3    user6@cluster2        export
End UserMap
```

On cluster2:

```
Begin UserMap
LOCAL    REMOTE                DIRECTION
user2    user1@cluster1        import
user6    user3@cluster1        import
End UserMap
```

Cluster1 configures user1 to run jobs as user2 and user3 to run jobs as user6.

Cluster2 configures user1 to run jobs as user2 and user3 to run jobs as user6.

Automatic time-based configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in `lsb.users` by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the **admin reconfig** command.

The expressions are evaluated by LSF every 10 minutes based on `mbatchd` start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting `mbatchd`, providing continuous system availability.

Example

From 12 - 1 p.m. daily (in Eastern Daylight Time), user `smith` has 10 job slots, but during other hours, user has only 5 job slots.

```
Begin User
USER_NAME  MAX_JOBS  JL/P
#if time(12-13 EDT)
smith      10        -
#else
smith      5         -
default    1         -
#endif
End User
```

Specifying the time zone is optional. If you do not specify a time zone, LSF uses the local system time zone. LSF supports all standard time zone abbreviations.

lsf.acct

The `lsf.acct` file is the LSF task log file.

The LSF Remote Execution Server, RES (see `res(8)`), generates a record for each task completion or failure. If the RES task logging is turned on (see `lsadmin(8)`), it appends the record to the task log file `lsf.acct.<host_name>`.

lsf.acct structure

The task log file is an ASCII file with one task record per line. The fields of each record are separated by blanks. The location of the file is determined by the LSF_RES_ACCTDIR variable defined in `lsf.conf`. If this variable is not defined, or the RES cannot access the log directory, the log file is created in `/tmp` instead.

Fields

The fields in a task record are ordered in the following sequence:

pid (%d)

Process ID for the remote task

userName (%s)

User name of the submitter

exitStatus (%d)

Task exit status

dispTime (%ld)

Dispatch time – time at which the task was dispatched for execution

termTime (%ld)

Completion time – time when task is completed/failed

fromHost (%s)

Submission host name

execHost (%s)

Execution host name

cwd (%s)

Current working directory

cmdln (%s)

Command line of the task

lsfRusage

The following fields contain resource usage information for the job (see `getrusage(2)`). If the value of some field is unavailable (due to job exit or the difference among the operating systems), -1 will be logged. Times are measured in seconds, and sizes are measured in KB.

ru_utime (%f)

User time used

ru_stime (%f)

System time used

ru_maxrss (%f)

Maximum shared text size

ru_ixrss (%f)

Integral of the shared text size over time (in KB seconds)

ru_ismrss (%f)

Integral of the shared memory size over time (valid only on Ultrix)

ru_idrss (%f)

Integral of the unshared data size over time

ru_isrss (%f)

Integral of the unshared stack size over time

ru_minflt (%f)

Number of page reclaims

ru_majflt (%f)

Number of page faults

ru_nswap (%f)

Number of times the process was swapped out

ru_inblock (%f)

Number of block input operations

ru_oublock (%f)

Number of block output operations

ru_ioch (%f)

Number of characters read and written (valid only on HP-UX)

ru_msgsnd (%f)

Number of System V IPC messages sent

ru_msgrcv (%f)

Number of messages received

ru_nsignals (%f)

Number of signals received

ru_nvcsw (%f)

Number of voluntary context switches

ru_nivcsw (%f)

Number of involuntary context switches

ru_exutime (%f)

Exact user time used (valid only on ConvexOS)

lsf.cluster

The cluster configuration file. There is one file for each cluster, called `lsf.cluster.cluster_name`. The `cluster_name` suffix is the name of the cluster defined in the Cluster section of the `lsf.shared` file. All LSF hosts are listed in this file, along with the list of LSF administrators and the installed LSF features.

Changing lsf.cluster configuration

After changing the `lsf.cluster.cluster_name` file, run the following commands:

- **lsadmin reconfig** to reconfigure LIM
- **badmin mbdrestart** to restart `mbatchd`
- **lsadmin limrestart** to restart LIM (on all changed non-master hosts)

Location

This file is typically installed in the directory defined by **LSF_ENVDIR**.

Structure

The `lsf.cluster.cluster_name` file contains the following configuration sections:

- Parameters section
- ClusterAdmins section
- Host section
- ResourceMap section
- RemoteClusters section

Parameters section

About lsf.cluster

The `lsf.cluster.cluster_name` file contains two types of configuration information:

Cluster definition information

affects all LSF applications. Defines cluster administrators, hosts that make up the cluster, attributes of each individual host such as host type or host model, and resources using the names defined in `lsf.shared`.

LIM policy information

affects applications that rely on LIM job placement policy. Defines load sharing and job placement policies provided by LIM.

Parameters

- ADJUST_DURATION
- ELIM_ABORT_VALUE
- ELIM_POLL_INTERVAL
- ELIMARGS
- EXINTERVAL
- FLOAT_CLIENTS
- FLOAT_CLIENTS_ADDR_RANGE
- HOST_INACTIVITY_LIMIT
- LSF_ELIM_BLOCKTIME
- LSF_ELIM_DEBUG
- LSF_ELIM_RESTARTS
- LSF_HOST_ADDR_RANGE
- MASTER_INACTIVITY_LIMIT
- PROBE_TIMEOUT
- RETRY_LIMIT

ADJUST_DURATION

Syntax

ADJUST_DURATION=*integer*

Description

Integer reflecting a multiple of EXINTERVAL that controls the time period during which load adjustment is in effect.

The **lsplace** and **lsloadadj** commands artificially raise the load on a selected host. This increase in load decays linearly to 0 over time.

Default

3

ELIM_ABORT_VALUE**Syntax**

ELIM_ABORT_VALUE=*integer*

Description

Integer that triggers an abort for an ELIM.

Default

97 (triggers abort)

ELIM_POLL_INTERVAL**Syntax**

ELIM_POLL_INTERVAL=*seconds*

Description

Time interval, in seconds, that the LIM samples external load index information. If your **elim** executable is programmed to report values more frequently than every 5 seconds, set the **ELIM_POLL_INTERVAL** so that it samples information at a corresponding rate.

Valid values

0.001 to 5

Default

5 seconds

ELIMARGS**Syntax**

ELIMARGS=*cmd_line_args*

Description

Specifies command-line arguments required by an **elim** executable on startup. Used only when the external load indices feature is enabled.

Default

Undefined

EXINTERVAL**Syntax**

EXINTERVAL=*time_in_seconds*

Description

Time interval, in seconds, at which the LIM daemons exchange load information

On extremely busy hosts or networks, or in clusters with a large number of hosts, load may interfere with the periodic communication between LIM daemons. Setting EXINTERVAL to a longer interval can reduce network load and slightly improve reliability, at the cost of slower reaction to dynamic load changes.

Note that if you define the time interval as less than 5 seconds, LSF automatically resets it to 5 seconds.

Default

15 seconds

FLOAT_CLIENTS**Syntax**

FLOAT_CLIENTS=*number_of_floating_clients*

Description

Sets the maximum allowable size for floating clients in a cluster. If **FLOAT_CLIENTS** is not specified in `lsf.cluster.cluster_name`, the floating LSF client feature is disabled.

**CAUTION:**

When the LSF floating client feature is enabled, any host can submit jobs to the cluster. You can limit which hosts can be LSF floating clients with the parameter **FLOAT_CLIENTS_ADDR_RANGE** in the `lsf.cluster.cluster_name` file.

Default

Undefined

FLOAT_CLIENTS_ADDR_RANGE**Syntax**

FLOAT_CLIENTS_ADDR_RANGE=*IP_address ...*

Description

Optional. IP address or range of addresses of domains from which floating client hosts can submit requests. Multiple ranges can be defined, separated by spaces. The IP address can have either a dotted quad notation (IPv4) or IP Next Generation (IPv6) format. LSF supports both formats; you do not have to map IPv4 addresses to an IPv6 format.

Note:

To use IPv6 addresses, you must define the parameter **LSF_ENABLE_SUPPORT_IPV6** in `lsf.conf`.

If the value of **FLOAT_CLIENT_ADDR_RANGE** is undefined, there is no security and any hosts can be LSF floating clients.

If a value is defined, security is enabled. If there is an error in the configuration of this variable, by default, no hosts will be allowed to be LSF floating clients.

When this parameter is defined, client hosts that do not belong to the domain will be denied access.

If a requesting host belongs to an IP address that falls in the specified range, the host will be accepted to become a floating client.

IP addresses are separated by spaces, and considered "OR" alternatives.

If you define **FLOAT_CLIENT_ADDR_RANGE** with:

- No range specified, all IPv4 and IPv6 clients can submit requests.
- Only an IPv4 range specified, only IPv4 clients within the range can submit requests.
- Only an IPv6 range specified, only IPv6 clients within the range can submit requests.
- Both an IPv6 and IPv4 range specified, IPv6 and IPv4 clients within the ranges can submit requests.

The asterisk (*) character indicates any value is allowed.

The dash (-) character indicates an explicit range of values. For example 1-4 indicates 1,2,3,4 are allowed.

Open ranges such as *-30, or 10-*, are allowed.

If a range is specified with fewer fields than an IP address such as 10.161, it is considered as 10.161.*.*.

Address ranges are validated at configuration time so they must conform to the required format. If any address range is not in the correct format, no hosts will be accepted as LSF floating clients, and an error message will be logged in the LIM log.

This parameter is limited to 2048 characters.

For IPv6 addresses, the double colon symbol (: :) indicates multiple groups of 16-bits of zeros. You can also use (: :) to compress leading and trailing zeros in an address filter, as shown in the following example:

```
FLOAT_CLIENTS_ADDR_RANGE=1080::8:800:20fc:*
```

This definition allows hosts with addresses 1080:0:0:0:8:800:20fc:* (three leading zeros).

You cannot use the double colon (: :) more than once within an IP address. You cannot use a zero before or after (: :). For example, 1080:0::8:800:20fc:* is not a valid address.

Notes

After you configure **FLOAT_CLIENTS_ADDR_RANGE**, check the `lim.log.host_name` file to make sure this parameter is correctly set. If this parameter is not set or is wrong, this will be indicated in the log file.

Examples

```
FLOAT_CLIENTS_ADDR_RANGE=100
```

All IPv4 and IPv6 hosts with a domain address starting with 100 will be allowed access.

- To specify only IPv4 hosts, set the value to 100.*
- To specify only IPv6 hosts, set the value to 100:*

```
FLOAT_CLIENTS_ADDR_RANGE=100-110.34.1-10.4-56
```

All client hosts belonging to a domain with an address having the first number between 100 and 110, then 34, then a number between 1 and 10, then, a number between 4 and 56 will be allowed access. Example: 100.34.9.45, 100.34.1.4, 102.34.3.20, etc. No IPv6 hosts are allowed.

```
FLOAT_CLIENTS_ADDR_RANGE=100.172.1.13 100.*.30-54 124.24-*.1.*-34
```

All client hosts belonging to a domain with the address 100.172.1.13 will be allowed access. All client hosts belonging to domains starting with 100, then any number, then a range of 30 to 54 will be allowed access. All client hosts belonging to domains starting with 124, then from 24 onward, then 1, then from 0 to 34 will be allowed access. No IPv6 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE=12.23.45.*

All client hosts belonging to domains starting with 12.23.45 are allowed. No IPv6 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE=100.*43

The * character can only be used to indicate any value. In this example, an error will be inserted in the LIM log and no hosts will be accepted to become LSF floating clients. No IPv6 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE=100.*43 100.172.1.13

Although one correct address range is specified, because *43 is not correct format, the entire line is considered not valid. An error will be inserted in the LIM log and no hosts will be accepted to become LSF floating clients. No IPv6 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE = 3ffe

All client IPv6 hosts with a domain address starting with 3ffe will be allowed access. No IPv4 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE = 3ffe:ffe::88bb:*

Expands to 3ffe:ffe:0:0:0:0:88bb:*. All IPv6 client hosts belonging to domains starting with 3ffe:ffe::88bb:* are allowed. No IPv4 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE = 3ffe-4fff:ffe::88bb:aa-ff 12.23.45.*

All IPv6 client hosts belonging to domains starting with 3ffe up to 4fff, then ffe::88bb, and ending with aa up to ff are allowed. All IPv4 client hosts belonging to domains starting with 12.23.45 are allowed.

FLOAT_CLIENTS_ADDR_RANGE = 3ffe-*:ffe::88bb:*-ff

All IPv6 client hosts belonging to domains starting with 3ffe up to ffff and ending with 0 up to ff are allowed. No IPv4 hosts are allowed.

Default

Undefined. No security is enabled. Any host in any domain is allowed access to LSF floating clients.

See also

LSF_ENABLE_SUPPORT_IPV6

HOST_INACTIVITY_LIMIT

Syntax

HOST_INACTIVITY_LIMIT=*integer*

Description

Integer that is multiplied by EXINTERVAL, the time period you set for the communication between the master and slave LIMs to ensure all parties are functioning.

A slave LIM can send its load information any time from EXINTERVAL to (HOST_INACTIVITY_LIMIT-1)*EXINTERVAL seconds. A master LIM sends a master announce to each host at least every EXINTERVAL*(HOST_INACTIVITY_LIMIT-1) seconds.

The HOST_INACTIVITY_LIMIT must be greater than or equal to 2.

Increase or decrease the host inactivity limit to adjust for your tolerance for communication between master and slaves. For example, if you have hosts that frequently become inactive, decrease the host inactivity limit. Note that to get the right interval, you may also have to adjust your EXINTERVAL.

Default

5

LSF_ELIM_BLOCKTIME

Syntax

LSF_ELIM_BLOCKTIME=*seconds*

Description

UNIX only; used when the external load indices feature is enabled.

Maximum amount of time the master external load information manager (MELIM) waits for a complete load update string from an **elim** executable. After the time period specified by LSF_ELIM_BLOCKTIME, the MELIM writes the last string sent by an **elim** in the LIM log file (`lim.log.host_name`) and restarts the **elim**.

Defining LSF_ELIM_BLOCKTIME also triggers the MELIM to restart **elim** executables if the **elim** does not write a complete load update string within the time specified for LSF_ELIM_BLOCKTIME.

Valid values

Non-negative integers. For example, if your **elim** writes name-value pairs with 1 second intervals between them, and your **elim** reports 12 load indices, allow at least 12 seconds for the **elim** to finish writing the entire load update string. In this case, define LSF_ELIM_BLOCKTIME as 15 seconds or more.

A value of 0 indicates that the MELIM expects to receive the entire load string all at once.

If you comment out or delete LSF_ELIM_BLOCKTIME, the MELIM waits 2 seconds for a complete load update string.

Default

4 seconds

See also

LSF_ELIM_RESTARTS to limit how many times the ELIM can be restarted.

LSF_ELIM_DEBUG

Syntax

LSF_ELIM_DEBUG=*y*

Description

UNIX only; used when the external load indices feature is enabled.

When this parameter is set to *y*, all external load information received by the load information manager (LIM) from the master external load information manager (MELIM) is logged in the LIM log file (`lim.log.host_name`).

Defining LSF_ELIM_DEBUG also triggers the MELIM to restart **elim** executables if the **elim** does not write a complete load update string within the time specified for LSF_ELIM_BLOCKTIME.

Default

Undefined; external load information sent by an to the MELIM is not logged.

See also

LSF_ELIM_BLOCKTIME to configure how long LIM waits before restarting the ELIM.

LSF_ELIM_RESTARTS to limit how many times the ELIM can be restarted.

LSF_ELIM_RESTARTS**Syntax**

LSF_ELIM_RESTARTS=*integer*

Description

UNIX only; used when the external load indices feature is enabled.

Maximum number of times the master external load information manager (MELIM) can restart **elim** executables on a host. Defining this parameter prevents an ongoing restart loop in the case of a faulty **elim**. The MELIM waits the LSF_ELIM_BLOCKTIME to receive a complete load update string before restarting the **elim**. The MELIM does not restart any **elim** executables that exit with ELIM_ABORT_VALUE.

Important:

Either LSF_ELIM_BLOCKTIME or LSF_ELIM_DEBUG must also be defined; defining these parameters triggers the MELIM to restart **elim** executables.

Valid values

Non-negative integers.

Default

Undefined; the number of **elim** restarts is unlimited.

See also

LSF_ELIM_BLOCKTIME, LSF_ELIM_DEBUG

LSF_HOST_ADDR_RANGE**Syntax**

LSF_HOST_ADDR_RANGE=*IP_address ...*

Description

Identifies the range of IP addresses that are allowed to be LSF hosts that can be dynamically added to or removed from the cluster.

**CAUTION:**

To enable dynamically added hosts after installation, you must define LSF_HOST_ADDR_RANGE in lsf.cluster.cluster_name, and LSF_DYNAMIC_HOST_WAIT_TIME in lsf.conf. If you enable dynamic hosts during installation, you must define an IP address range after installation to enable security.

If a value is defined, security for dynamically adding and removing hosts is enabled, and only hosts with IP addresses within the specified range can be added to or removed from a cluster dynamically.

Specify an IP address or range of addresses, using either a dotted quad notation (IPv4) or IP Next Generation (IPv6) format. LSF supports both formats; you do not have to map IPv4 addresses to an IPv6 format. Multiple ranges can be defined, separated by spaces.

Note:

To use IPv6 addresses, you must define the parameter LSF_ENABLE_SUPPORT_IPV6 in lsf.conf.

If there is an error in the configuration of LSF_HOST_ADDR_RANGE (for example, an address range is not in the correct format), no host will be allowed to join the cluster dynamically and an error message will be

logged in the LIM log. Address ranges are validated at startup, reconfiguration, or restart, so they must conform to the required format.

If a requesting host belongs to an IP address that falls in the specified range, the host will be accepted to become a dynamic LSF host.

IP addresses are separated by spaces, and considered "OR" alternatives.

If you define the parameter LSF_HOST_ADDR_RANGE with:

- No range specified, all IPv4 and IPv6 clients are allowed.
- Only an IPv4 range specified, only IPv4 clients within the range are allowed.
- Only an IPv6 range specified, only IPv6 clients within the range are allowed.
- Both an IPv6 and IPv4 range specified, IPv6 and IPv4 clients within the ranges are allowed.

The asterisk (*) character indicates any value is allowed.

The dash (-) character indicates an explicit range of values. For example 1-4 indicates 1,2,3,4 are allowed.

Open ranges such as *-30, or 10-*, are allowed.

For IPv6 addresses, the double colon symbol (::) indicates multiple groups of 16-bits of zeros. You can also use (::) to compress leading and trailing zeros in an address filter, as shown in the following example:

```
LSF_HOST_ADDR_RANGE=1080::8:800:20fc:*
```

This definition allows hosts with addresses 1080:0:0:0:8:800:20fc:* (three leading zeros).

You cannot use the double colon (::) more than once within an IP address. You cannot use a zero before or after (::). For example, 1080:0::8:800:20fc:* is not a valid address.

If a range is specified with fewer fields than an IP address such as 10.161, it is considered as 10.161.*.*.

This parameter is limited to 2048 characters.

Notes

After you configure LSF_HOST_ADDR_RANGE, check the `lim.log.host_name` file to make sure this parameter is correctly set. If this parameter is not set or is wrong, this will be indicated in the log file.

Examples

```
LSF_HOST_ADDR_RANGE=100
```

All IPv4 and IPv6 hosts with a domain address starting with 100 will be allowed access.

- To specify only IPv4 hosts, set the value to `100.*`
- To specify only IPv6 hosts, set the value to `100:*`

```
LSF_HOST_ADDR_RANGE=100-110.34.1-10.4-56
```

All hosts belonging to a domain with an address having the first number between 100 and 110, then 34, then a number between 1 and 10, then, a number between 4 and 56 will be allowed access. No IPv6 hosts are allowed. Example: 100.34.9.45, 100.34.1.4, 102.34.3.20, etc.

```
LSF_HOST_ADDR_RANGE=100.172.1.13 100.*.30-54 124.24-*.1.*-34
```

The host with the address 100.172.1.13 will be allowed access. All hosts belonging to domains starting with 100, then any number, then a range of 30 to 54 will be allowed access. All hosts belonging to domains starting with 124, then from 24 onward, then 1, then from 0 to 34 will be allowed access. No IPv6 hosts are allowed.

```
LSF_HOST_ADDR_RANGE=12.23.45.*
```

All hosts belonging to domains starting with 12.23.45 are allowed. No IPv6 hosts are allowed.

```
LSF_HOST_ADDR_RANGE=100.*43
```

The * character can only be used to indicate any value. The format of this example is not correct, and an error will be inserted in the LIM log and no hosts will be able to join the cluster dynamically. No IPv6 hosts are allowed.

```
LSF_HOST_ADDR_RANGE=100.*43 100.172.1.13
```

Although one correct address range is specified, because *43 is not correct format, the entire line is considered not valid. An error will be inserted in the LIM log and no hosts will be able to join the cluster dynamically. No IPv6 hosts are allowed.

```
LSF_HOST_ADDR_RANGE = 3ffe
```

All client IPv6 hosts with a domain address starting with 3ffe will be allowed access. No IPv4 hosts are allowed.

```
LSF_HOST_ADDR_RANGE = 3ffe:ffe::88bb:*
```

Expands to 3ffe:ffe:0:0:0:0:88bb:*. All IPv6 client hosts belonging to domains starting with 3ffe:ffe::88bb:* are allowed. No IPv4 hosts are allowed.

```
LSF_HOST_ADDR_RANGE = 3ffe-4fff:ffe::88bb:aa-ff 12.23.45.*
```

All IPv6 client hosts belonging to domains starting with 3ffe up to 4fff, then ffe::88bb, and ending with aa up to ff are allowed. IPv4 client hosts belonging to domains starting with 12.23.45 are allowed.

```
LSF_HOST_ADDR_RANGE = 3ffe-*:ffe::88bb:*-ff
```

All IPv6 client hosts belonging to domains starting with 3ffe up to ffff and ending with 0 up to ff are allowed. No IPv4 hosts are allowed.

Default

Undefined (dynamic host feature disabled). If you enable dynamic hosts during installation, no security is enabled and all hosts can join the cluster.

See also

LSF_ENABLE_SUPPORT_IPV6

MASTER_INACTIVITY_LIMIT

Syntax

```
MASTER_INACTIVITY_LIMIT=integer
```

Description

An integer reflecting a multiple of EXINTERVAL. A slave will attempt to become master if it does not hear from the previous master after (HOST_INACTIVITY_LIMIT + *host_number**MASTER_INACTIVITY_LIMIT)*EXINTERVAL seconds, where *host_number* is the position of the host in `lsf.cluster.cluster_name`.

The master host is *host_number* 0.

Default

2

PROBE_TIMEOUT

Syntax

```
PROBE_TIMEOUT=time_in_seconds
```

Description

Specifies the timeout in seconds to be used for the connect(2) system call

Before taking over as the master, a slave LIM will try to connect to the last known master via TCP.

Default

2 seconds

RETRY_LIMIT**Syntax**

RETRY_LIMIT=*integer*

Description

Integer reflecting a multiple of EXINTERVAL that controls the number of retries a master or slave LIM makes before assuming that the slave or master is unavailable.

If the master does not hear from a slave for HOST_INACTIVITY_LIMIT exchange intervals, it will actively poll the slave for RETRY_LIMIT exchange intervals before it will declare the slave as unavailable. If a slave does not hear from the master for HOST_INACTIVITY_LIMIT exchange intervals, it will actively poll the master for RETRY_LIMIT intervals before assuming that the master is down.

Default

2

ClusterAdmins section

(Optional) The ClusterAdmins section defines the LSF administrators for the cluster. The only keyword is ADMINISTRATORS.

If the ClusterAdmins section is not present, the default LSF administrator is root. Using root as the primary LSF administrator is not recommended.

ADMINISTRATORS**Syntax**

ADMINISTRATORS=*administrator_name ...*

Description

Specify UNIX user names.

You can also specify UNIX user group names, Windows user names, and Windows user group names. To specify a Windows user account or user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_name* or *DOMAIN_NAME\user_group*).

The first administrator of the expanded list is considered the primary LSF administrator. The primary administrator is the owner of the LSF configuration files, as well as the working files under *LSB_SHARED_DIR/cluster_name*. If the primary administrator is changed, make sure the owner of the configuration files and the files under *LSB_SHARED_DIR/cluster_name* are changed as well.

Administrators other than the primary LSF administrator have the same privileges as the primary LSF administrator except that they do not have permission to change LSF configuration files. They can perform clusterwide operations on jobs, queues, or hosts in the system.

For flexibility, each cluster may have its own LSF administrators, identified by a user name, although the same administrators can be responsible for several clusters.

Use the `-l` option of the **lsclusters** command to display all of the administrators within a cluster.

Windows domain:

- If the specified user or user group is a domain administrator, member of the `Power Users` group or a group with domain administrative privileges, the specified user or user group must belong to the LSF user domain.
- If the specified user or user group is a user or user group with a lower degree of privileges than outlined in the previous point, the user or user group must belong to the LSF user domain and be part of the `Global Admins` group.

Windows workgroup

- If the specified user or user group is not a workgroup administrator, member of the `Power Users` group, or a group with administrative privileges on each host, the specified user or user group must belong to the `Local Admins` group on each host.

Compatibility

For backwards compatibility, `ClusterManager` and `Manager` are synonyms for `ClusterAdmins` and `ADMINISTRATORS` respectively. It is possible to have both sections present in the same `lsf.cluster.cluster_name` file to allow daemons from different LSF versions to share the same file.

Example

The following gives an example of a cluster with two LSF administrators. The user listed first, `user2`, is the primary administrator.

```
Begin ClusterAdmins
ADMINISTRATORS = user2 user7
End ClusterAdmins
```

Default

lsfadmin

Host section

The Host section is the last section in `lsf.cluster.cluster_name` and is the only required section. It lists all the hosts in the cluster and gives configuration information for each host.

The order in which the hosts are listed in this section is important, because the first host listed becomes the LSF master host. Since the master LIM makes all placement decisions for the cluster, it should be on a fast machine.

The LIM on the first host listed becomes the master LIM if this host is up; otherwise, that on the second becomes the master if its host is up, and so on. Also, to avoid the delays involved in switching masters if the first machine goes down, the master should be on a reliable machine. It is desirable to arrange the list such that the first few hosts in the list are always in the same subnet. This avoids a situation where the second host takes over as master when there are communication problems between subnets.

Example Host section

This example Host section contains descriptive information for three hosts:

```
Begin Host
HOSTNAME  model  type  server  RESOURCES  RUNWINDOW
hostA     SparcIPC  Sparc  1      (sunos frame)  ()
hostD     Sparc10  Sparc  1      (sunos)        (5:18:30-1:8:30)
hostD     !        !      1      ()              ()
hostE     !        !      1      (linux !bigmem)  ()
End Host
```

Descriptive fields

The following fields are required in the Host section:

- HOSTNAME
- RESOURCES
- type
- model

The following fields are optional:

- server
- nd
- RUNWINDOW
- REXPRI

HOSTNAME**Description**

Official name of the host as returned by `hostname(1)`

The name must be listed in `lsf.shared` as belonging to this cluster.

Pattern definition

You can use string literals and special characters when defining host names. Each entry cannot contain any spaces, as the list itself is space delimited.

You can use the following special characters to specify hosts:

- Use square brackets with a hyphen (`[integer1-integer2]`) or a comma (`[integer1:integer2]`) to define a range of non-negative integers anywhere in the host name. The first integer must be less than the second integer.
- Use square brackets with commas (`[integer1, integer2 ...]`) to define individual non-negative integers anywhere in the host name.
- Use square brackets with commas and hyphens or colons (for example, `[integer1-integer2, integer3, integer4:integer5, integer6:integer7]`) to define different ranges of non-negative integers anywhere in the host name.
- Use multiple sets of square brackets (with the supported special characters) to define multiple sets of non-negative integers anywhere in the host name. For example, `hostA[1,3]B[1-3]` includes `hostA1B1`, `hostA1B2`, `hostA1B3`, `hostA3B1`, `hostA3B2`, and `hostA3B3`.

model**Description**

Host model

The name must be defined in the HostModel section of `lsf.shared`. This determines the CPU speed scaling factor applied in load and placement calculations.

Optionally, the `!` keyword for the model or type column, indicates that the host model or type is to be automatically detected by the LIM running on the host.

nd**Description**

Number of local disks

lsf.cluster

This corresponds to the ndisks static resource. On most host types, LSF automatically determines the number of disks, and the nd parameter is ignored.

nd should only count local disks with file systems on them. Do not count either disks used only for swapping or disks mounted with NFS.

Default

The number of disks determined by the LIM, or 1 if the LIM cannot determine this

RESOURCES

Description

The static Boolean resources and static or dynamic numeric and string resources available on this host.

The resource names are strings defined in the Resource section of `lsf.shared`. You may list any number of resources, enclosed in parentheses and separated by blanks or tabs. For example:

```
(fs frame hpux)
```

Optionally, you can specify an exclusive resource by prefixing the resource with an exclamation mark (!). For example, resource `bigmem` is defined in `lsf.shared`, and is defined as an exclusive resource for `hostE`:

```
Begin Host
HOSTNAME  model  type  server RESOURCES          RUNWINDOW
...
hostE     !      !      1      (linux !bigmem)      ()
...
End Host
```

Square brackets are not valid and the resource name must be alphanumeric.

You must explicitly specify the exclusive resources in the resource requirements for the job to select a host with an exclusive resource for a job. For example:

```
bsub -R "bigmem" myjob
```

or

```
bsub -R "defined(bigmem)" myjob
```

You can specify static and dynamic numeric and string resources in the resource column of the Host clause. For example:

```
Begin Host
HOSTNAME  model  type  server RESOURCES  #Keywords
hostA     !      !      1      (mg elimres patchrev=3 owner=user1)
hostB     !      !      1      (specman=5 switch=1 owner=test)
hostC     !      !      1      (switch=2 rack=rack2_2_3 owner=test)
hostD     !      !      1      (switch=1 rack=rack2_2_3 owner=test)
End Host
```

Static resource information is displayed by **lshosts**, with exclusive resources prefixed by and exclamation mark (!).

REXPRI

Description

UNIX only

Default execution priority for interactive remote jobs run under the RES

The range is from -20 to 20. REXPRI corresponds to the BSD-style nice value used for remote jobs. For hosts with System V-style nice values with the range 0 - 39, a REXPRI of -20 corresponds to a nice value of 0, and +20 corresponds to 39. Higher values of REXPRI correspond to lower execution priority; -20 gives the highest priority, 0 is the default priority for login sessions, and +20 is the lowest priority.

Default

0

RUNWINDOW

Description

Dispatch window for interactive tasks.

When the host is not available for remote execution, the host status is `lockW` (locked by run window). LIM does not schedule interactive tasks on hosts locked by dispatch windows. Run windows only apply to interactive tasks placed by LIM. The LSF batch system uses its own (optional) host dispatch windows to control batch job processing on batch server hosts.

Format

A dispatch window consists of one or more time windows in the format *begin_time-end_time*. No blanks can separate *begin_time* and *end_time*. Time is specified in the form *[day:]hour[:minute]*. If only one field is specified, LSF assumes it is an *hour*. Two fields are assumed to be *hour:minute*. Use blanks to separate time windows.

Default

Always accept remote jobs

server

Description

Indicates whether the host can receive jobs from other hosts

Specify 1 if the host can receive jobs from other hosts; specify 0 otherwise. Servers that are set to 0 are LSF clients. Client hosts do not run the LSF daemons. Client hosts can submit interactive and batch jobs to the cluster, but they cannot execute jobs sent from other hosts.

Default

1

type

Description

Host type as defined in the HostType section of `lsf.shared`

The strings used for host types are determined by the system administrator; for example, SUNSOL, DEC, or HPPA. The host type is used to identify binary-compatible hosts.

The host type is used as the default resource requirement. That is, if no resource requirement is specified in a placement request, the task is run on a host of the same type as the sending host.

Often one host type can be used for many machine models. For example, the host type name SUNSOL6 might be used for any computer with a SPARC processor running SunOS 6. This would include many Sun models and quite a few from other vendors as well.

Optionally, the `!` keyword for the model or type column, indicates that the host model or type is to be automatically detected by the LIM running on the host.

ResourceMap section

The ResourceMap section defines shared resources in your cluster. This section specifies the mapping between shared resources and their sharing hosts. When you define resources in the Resources section of `lsf.shared`, there is no distinction between a shared and non-shared resource. By default, all resources are not shared and are local to each host. By defining the ResourceMap section, you can define resources that are shared by all hosts in the cluster or define resources that are shared by only some of the hosts in the cluster.

This section must appear after the Host section of `lsf.cluster.cluster_name`, because it has a dependency on host names defined in the Host section.

ResourceMap section structure

The first line consists of the keywords RESOURCENAME and LOCATION. Subsequent lines describe the hosts that are associated with each configured resource.

Example ResourceMap section

```
Begin ResourceMap
RESOURCENAME  LOCATION
verilog      (5@[all])
local        ([host1 host2] [others])
End ResourceMap
```

The resource `verilog` must already be defined in the RESOURCE section of the `lsf.shared` file. It is a static numeric resource shared by all hosts. The value for `verilog` is 5. The resource `local` is a numeric shared resource that contains two instances in the cluster. The first instance is shared by two machines, `host1` and `host2`. The second instance is shared by all other hosts.

Resources defined in the ResourceMap section can be viewed by using the `-s` option of the **lshosts** (for static resource) and **lsload** (for dynamic resource) commands.

LOCATION

Description

Defines the hosts that share the resource

For a static resource, you must define an initial value here as well. Do not define a value for a dynamic resource.

instance is a list of host names that share an instance of the resource. The reserved words `all`, `others`, and `default` can be specified for the instance:

`all` - Indicates that there is only one instance of the resource in the whole cluster and that this resource is shared by all of the hosts

Use the not operator (`~`) to exclude hosts from the `all` specification. For example:

```
(2@[all ~host3 ~host4])
```

This means that 2 units of the resource are shared by all server hosts in the cluster made up of `host1`, `host2`, . . . `hostn`, except for `host3` and `host4`. This is useful if you have a large cluster but only want to exclude a few hosts.

The parentheses are required in the specification. The not operator can only be used with the `all` keyword. It is not valid with the keywords `others` and `default`.

`others` - Indicates that the rest of the server hosts not explicitly listed in the LOCATION field comprise one instance of the resource

For example:

```
2@[host1] 4@[others]
```

This indicates that there are 2 units of the resource on host1 and 4 units of the resource shared by all other hosts.

The default keyword indicates an instance of a resource on each host in the cluster

This specifies a special case where the resource is in effect not shared and is local to every host. default means at each host. Normally, you should not need to use default, because by default, all built-in resources are local to each host. However, resources that you defined must always be mapped. You might want to use ResourceMap for a non-shared static resource if you need to specify different values for the resource on different hosts.

RESOURCENAME

Description

Name of the resource

This resource name must be defined in the Resource section of the `lsf.shared` file. You must specify at least a name and description for the resource, using the keywords RESOURCENAME and DESCRIPTION.

- A resource name cannot begin with a number
- A resource name cannot contain any of the following characters:

```
: . ( ) [ + - * / ! & | < > @ =
```

- A resource name cannot be any of the following reserved names:

```
cpu cpuf io logins ls idle maxmem maxswp maxtmp type model status it
mem ncpus define_ncpus_cores define_ncpus_procs
define_ncpus_threads ndisks pg r15m r15s r1m swap swp tmp ut
```

- To avoid conflict with `inf` and `nan` keywords in 3rd-party libraries, resource names should not begin with `inf` or `nan` (upper case or lower case). Resource requirement strings, such as `-R "infra"` or `-R "nano"` will cause an error. Use `-R "defined(infixx)"` or `-R "defined(nanxx)"`, to specify these resource names.
- Resource names are case sensitive
- Resource names can be up to 39 characters in length

RemoteClusters section

Optional. This section is used only in a MultiCluster environment. By default, the local cluster can obtain information about all other clusters specified in `lsf.shared`. The RemoteClusters section limits the clusters that the local cluster can obtain information about.

The RemoteClusters section is required if you want to configure cluster equivalency, cache interval, daemon authentication across clusters, or if you want to run parallel jobs across clusters. To maintain compatibility in this case, make sure the list includes all clusters specified in `lsf.shared`, even if you only configure the default behavior for some of the clusters.

The first line consists of keywords. CLUSTERNAME is mandatory and the other parameters are optional.

Subsequent lines configure the remote cluster.

Example RemoteClusters section

```
Begin RemoteClusters
CLUSTERNAME  EQUIV  CACHE_INTERVAL  RECV_FROM  AUTH
cluster1     Y        60                Y          KRB
cluster2     N        60                Y          -
cluster4     N        60                N          PKI
End RemoteClusters
```

CLUSTERNAME**Description**

Remote cluster name

Defines the Remote Cluster list. Specify the clusters you want the local cluster to recognize. Recognized clusters must also be defined in `lsf.shared`. Additional clusters listed in `lsf.shared` but not listed here will be ignored by this cluster.

EQUIV**Description**

Specify 'Y' to make the remote cluster equivalent to the local cluster. Otherwise, specify N. The master LIM considers all equivalent clusters when servicing requests from clients for load, host, or placement information.

EQUIV changes the default behavior of LSF commands and utilities and causes them to automatically return load (**lsload**), host (**lshosts**), or placement (**lsplace**) information about the remote cluster as well as the local cluster, even when you don't specify a cluster name.

CACHE_INTERVAL**Description**

Specify the load information cache threshold, in seconds. The host information threshold is twice the value of the load information threshold.

To reduce overhead and avoid updating information from remote clusters unnecessarily, LSF displays information in the cache, unless the information in the cache is older than the threshold value.

Default

60 seconds

RECV_FROM**Description**

Specifies whether the local cluster accepts parallel tasks that originate in a remote cluster. RECV_FROM does not affect regular or interactive batch jobs.

Specify Y if you want to run parallel jobs across clusters. Otherwise, specify N.

Default

Y

AUTH**Description**

Defines the preferred authentication method for LSF daemons communicating across clusters. Specify the same method name that is used to identify the corresponding **eauth** program (**eauth.method_name**). If the remote cluster does not prefer the same method, LSF uses default security between the two clusters.

Default

- (only privileged port (**setuid**) authentication is used between clusters)

lsf.conf

The `lsf.conf` file controls the operation of LSF. The `lsf.conf` file is created during installation and records all the settings chosen when LSF was installed. The `lsf.conf` file dictates the location of the specific configuration files and operation of individual servers and applications.

About lsf.conf

The `lsf.conf` file is used by LSF and applications built on top of it. For example, information in the `lsf.conf` file is used by LSF daemons and commands to locate other configuration files, executables, and network services. The `lsf.conf` file is updated, if necessary, when you upgrade to a new version.

This file can also be expanded to include application-specific parameters.

Parameters in this file can also be set as environment variables, except for the parameters related to job packs.

If you have installed LSF in a mixed Windows and UNIX/Linux cluster, you must make sure that `lsf.conf` parameters set on UNIX and Linux match any corresponding parameters in the local `lsf.conf` files on your Windows hosts.

Corresponding parameters in ego.conf

When EGO is enabled in LSF, you can configure some LSF parameters in the `lsf.conf` file that have corresponding EGO parameter names in `EGO_CONFDIR/ego.conf` (`LSF_CONFDIR/lsf.conf` is a separate file from `EGO_CONFDIR/ego.conf`). If both the LSF and the EGO parameters are set in their respective files, the definition in `ego.conf` is used. You must continue to set LSF parameters only in `lsf.conf`.

When EGO is enabled in the LSF cluster (**LSF_ENABLE_EGO=Y**), you also can set the following EGO parameters related to LIM, PIM, and ELIM in either `lsf.conf` or `ego.conf`:

- `EGO_DISABLE_UNRESOLVABLE_HOST` (dynamically added hosts only)
- `EGO_ENABLE_AUTO_DAEMON_SHUTDOWN`
- `EGO_DAEMONS_CPUS`
- `EGO_DEFINE_NCPUS`
- `EGO_SLAVE_CTRL_REMOTE_HOST`
- `EGO_WORKDIR`
- `EGO_PIM_SWAP_REPORT`
- `EGO_ESLIM_TIMEOUT`

If EGO is not enabled, you do not need to set these parameters.

See *Administering IBM Spectrum LSF* for more information about configuring LSF for EGO.

After you change lsf.conf configuration

Depending on which parameters you change, you might need to run one or more of the following commands after you change parameters in the `lsf.conf` file:

- **lsadmin reconfig** to reconfigure LIM
- **badmin reconfig** to reconfigure `mbatchd` without restarting
- **badmin mbdrestart** to restart `mbatchd`
- **badmin hrestart** to restart `sbatchd`

Note: Most parameters require only reconfiguration (**lsadmin reconfig** or **badmin reconfig**). Several LSF parameters require you to restart `mbatchd` (**badmin mbdrestart**). Parameters that require restart are indicated in the parameter description.

Location

The default location of the `lsf.conf` file is in the `$LSF_TOP/conf` directory. This default location can be overridden when necessary by either the environment variable **LSF_ENVDIR**.

Format

Each entry in `lsf.conf` has one of the following forms:

```
NAME=VALUE
NAME=
NAME="STRING1 STRING2 ..."
```

The equal sign `=` must follow each NAME even if no value follows and there should be no space beside the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Lines starting with a pound sign (`#`) are comments and are ignored. Do not use `#if` as this is reserved syntax for time-based configuration.

DAEMON_SHUTDOWN_DELAY**Syntax**

`DAEMON_SHUTDOWN_DELAY=time_in_seconds`

Description

Applies when **EGO_ENABLE_AUTO_DAEMON_SHUTDOWN=Y**. Controls amount of time the slave LIM waits to communicate with other (RES and SBD) local daemons before exiting. Used to shorten or lengthen the time interval between a host attempting to join the cluster and, if it was unsuccessful, all of the local daemons shutting down.

The value should not be less than the minimum interval of RES and SBD housekeeping. Most administrators should set this value to somewhere between 3 minutes and 60 minutes.

Default

Set to 180 seconds (3 minutes) at time of installation for the DEFAULT configuration template. If otherwise undefined, then 1800 seconds (30 minutes).

EBROKERD_HOST_CLEAN_DELAY

For LSF resource connector. Specifies the delay, in minutes, after which the **ebrokerd** daemon removes information about relinquished or reclaimed hosts. This parameter allows the **bhosts -rc** and **bhosts -rconly** command options to get LSF resource connector provider host information for some time after they are deprovisioned.

Syntax

`EBROKERD_HOST_CLEAN_DELAY=minutes`

Description

After configuring this parameter, run **badmin mbdrestart** on the master host to restart **ebrokerd**.

After configuring this parameter, run **lsadmin limrestart** on the master host and, if any of the specified host names is not the master, **lsadmin limrestart *host_name***.

Example

```
EBROKERD_HOST_CLEAN_DELAY=30
```

Default

60 minutes

See also

- **LSF_MQ_BROKER_HOSTS**
- **MQTT_BROKER_HOST**
- **MQTT_BROKER_PORT**
- **bhosts -rc** and **bhosts -rconly**

EGO_DAEMONS_CPUS**Syntax**

EGO_DAEMONS_CPUS=*lim_cpu_list*

lim_cpu_list

Defines the list of master host CPUS where the lim daemon processes can run (hard CPU affinity).
Format the list as a white-space delimited list of CPU numbers.

Description

By default, **lim** can run on any CPUs. If EGO_DAEMONS_CPUS is set, LIM only run on a specified list of CPUs. An empty list means that LIM can run on any CPUs. Use spaces to separate multiple CPUs.

The operating system can assign other processes to run on the same CPU; however, if utilization of the bound CPU is lower than utilization of the unbound CPUs.

Related parameters

To improve scheduling and dispatch performance of all LSF daemons, you should use EGO_DAEMONS_CPUS together with LSF_DAEMONS_CPUS (in `ego.conf` or `lsf.conf`), which controls **mbatchd** and **mbschd** CPU allocation, and MBD_QUERY_CPUS, which binds **mbatchd** query processes to specific CPUs so that higher priority daemon processes can run more efficiently. To get best performance, CPU allocation for all four daemons should be assigned their own CPUs. For example, on a 4 CPU SMP host, the following configuration gives the best performance:

```
EGO_DAEMONS_CPUS=0 LSF_DAEMONS_CPUS=1:2 MBD_QUERY_CPUS=3
```

Examples

If you specify

```
EGO_DAEMONS_CPUS=1
```

the LIM processes run only on CPU number 1 on the master host.

Important

You can specify CPU affinity only for master hosts that use one of the following operating systems:

- Linux 2.6 or higher
- Solaris 10 or higher

Default

Not defined

See also

LSF_DAEMONS_CPUS and **MBD_QUERY_CPUS** in `lsb.params`

EGO_DEFINE_NCPUS

Syntax

`EGO_DEFINE_NCPUS=procs | cores | threads`

Description

If defined, enables an administrator to define a value other than the number of cores available for `ncpus`. The value of `ncpus` depends on the value of `EGO_DEFINE_NCPUS` as follows:

EGO_DEFINE_NCPUS=procs

`ncpus`=number of processors

EGO_DEFINE_NCPUS=cores

`ncpus`=number of processors x number of cores

EGO_DEFINE_NCPUS=threads

`ncpus`=number of processors x number of cores x number of threads

Note: When **PARALLEL_SCHED_BY_SLOT=Y** in `lsb.params`, the resource requirement string keyword `ncpus` refers to the number of slots instead of the number of CPUs, however **lshosts** output will continue to show `ncpus` as defined by **EGO_DEFINE_NCPUS** in `lsf.conf`.

For changes to **EGO_DEFINE_NCPUS** to take effect, restart **lim**, **mbatchd**, and **sbatchd** daemons in sequence.

Default

`EGO_DEFINE_NCPUS=cores`

EGO_ELIM_DIR

Syntax

`EGO_ELIM_DIR=directory`

Description

Windows only. Directory in which the EGO **elim** binary files are installed.

Specify a file path and `melim` uses this file path to find and use the following **elim** files: `elim.bat`, `elim.cmd`, `elim.vbs`, and `elim.sh`.

This parameter allows you to specify a separate location for **elim** files that is different from the **LSF_SERVERDIR** parameter, such as a shared file system.

Default

`LSF_SERVERDIR`

See also

LSF_SERVERDIR, **LSF_ELIM_DIR**

EGO_ENABLE_AUTO_DAEMON_SHUTDOWN

Syntax

EGO_ENABLE_AUTO_DAEMON_SHUTDOWN="Y" | "N"

Description

For hosts that attempted to join the cluster but failed to communicate within the LSF_DYNAMIC_HOST_WAIT_TIME period, automatically shuts down any running daemons.

This parameter can be useful if an administrator remove machines from the cluster regularly (by editing `lsf.cluster` file) or when a host belonging to the cluster is imaged, but the new host should not be part of the cluster. An administrator no longer has to go to each host that is not a part of the cluster to shut down any running daemons.

Default

Set to Y at time of installation. If otherwise undefined, then N.

EGO_PARAMETER

EGO_ENABLE_AUTO_DAEMON_SHUTDOWN

EGO_ESLIM_TIMEOUT

Syntax

EGO_ESLIM_TIMEOUT=*time_seconds*

Description

Controls how long the LIM waits for any external static LIM scripts to run. After the timeout period expires, the LIM stops the scripts.

Use the external static LIM to automatically detect the operating system type and version of hosts.

LSF automatically detects the operating systems types and versions and displays them when running **lshosts -l** or **lshosts -s**. You can then specify those types in any **-R** resource requirement string. For example, `bsub -R "select[ostype=RHEL4.6]"`.

Default

10 seconds

EGO_PARAMETER

EGO_ESLIM_TIMEOUT

EGO_LIM_PG_SMOOTH_FACTOR

Syntax

EGO_LIM_PG_SMOOTH_FACTOR=*smoothing_factor*|0

Description

This parameter is only supported in Linux platforms.

Specifies the smoothing factor when lim reports the host page rate. The smoothing factor controls how fast reported values converge to an instantaneous value.

Valid values

smoothing_factor must be an integer between 0 and 10.

If set to 0, no smoothing is applied and the reported value is equal to the instantaneous value. For other integers, the larger the value, the more time LSF needs to react to page rate change in the host.

Default

4

EGO_PIM_SWAP_REPORT

Enables PIM to report swap space usage for job processes.

Syntax

EGO_PIM_SWAP_REPORT=Y|y|N|n

Description

By default, LSF collects both memory and swap usage through PIM:

- If the **EGO_PIM_SWAP_REPORT=n** parameter is set in the `lsf.conf` file (this is the default), swap usage is virtual memory (VSZ) of the entire job process.
- If the **EGO_PIM_SWAP_REPORT=y** parameter is set in the `lsf.conf` file, the resident set size (RSS) is subtracted from the virtual memory usage. RSS is the portion of memory occupied by a process that is held in main memory. Swap usage is collected as the $VSZ - RSS$.

For example, if a job process allocates memory as follows,

```
malloc(p, 200MB);
memset(p, 150MB);
```

then **bjobs -l** shows SWAP as either 200 MB if **EGO_PIM_SWAP_REPORT=n**, or 50 MB if **EGO_PIM_SWAP_REPORT=y**.

Default

N

JOB_STARTER_EXTEND**Syntax**

JOB_STARTER_EXTEND="preservestarter" | "preservestarter userstarter"

Description

Applies to Windows execution hosts only.

Allows you to use a job starter that includes symbols (for example: &&, |, | |). The job starter configured in **JOB_STARTER_EXTEND** can handle these special characters. The file `$LSF_TOP/10.1/misc/examples/preservestarter.c` is the only extended job starter created by default. Users can also develop their own extended job starters based on `preservestarter.c`.

You must also set `JOB_STARTER=preservestarter` in `lsb.queues`.

Default

Not defined.

LS_DUPLICATE_RESOURCE_CHECKING

Syntax

LS_DUPLICATE_RESOURCE_CHECKING=Y|N

Description

When the parameter is enabled, there are strict checks for types of duplicated LS resources. LS resources can only override the LSF resources which have the same necessary characteristics (shared, numeric, non-built-in) as the LS resources. Otherwise, the LSF resource will take effect and the LS resource will be ignored.

When the parameter is disabled, there are no checks for duplicated LS resources. LS resources can override the resource in LSF without considering the resource's characteristics. In such cases, LSF may schedule jobs incorrectly. It is not recommended to define duplicated resources in both LS and LSF.

After changing the parameter value, run **admin mbdrestart**.

Default

Y

LSB_AFS_BIN_DIR

Syntax

LSB_AFS_BIN_DIR=path to aklog directory

Description

If **LSB_AFS_JOB_SUPPORT=Y**, then LSF will need **aklog** in AFS to create a new PAG and apply for an AFS token. You can then use **LSB_AFS_BIN_DIR** to tell LSF the file path and directory where **aklog** resides.

If **LSB_AFS_BIN_DIR** is not defined, LSF will search in the following order: /bin, /usr/bin, /usr/local/bin, /usr/afs/bin. The search stops as soon as an executable **aklog** is found.

Default

Not defined.

LSB_AFS_JOB_SUPPORT

Syntax

LSB_AFS_JOB_SUPPORT=Y|y|N|n

Description

When this parameter is set to Y|y:

1. LSF assumes the user's job is running in an AFS environment, and calls **aklog -setpag** to create a new PAG for the user's job if it is a sequential job, or to create a separate PAG for each task res if the job is a **blaunch** job.
2. LSF runs the **erenew** script after the TGT is renewed. This script is primarily used to run **aklog**.
3. LSF assumes that **JOB_SPOOL_DIR** resides in the AFS volume. It kerberizes the child **sbatchd** to get the AFS token so the child **sbatchd** can access **JOB_SPOOL_DIR**.

If this parameter changed, restart root res to make it take effect.

Default

N|n. LSF does not perform the three steps mentioned above.

LSB_AFS_LIB_DIR

Syntax

LSB_AFS_LIB_DIR=list of directories

Description

If **LSB_AFS_JOB_SUPPORT=Y**, LSF will need `libkopenafs.so` or `libkopenafs.so.1` to create PAG. You can then use **LSB_AFS_LIB_DIR** to tell LSF the directory in which `libkopenafs.so` or `libkopenafs.so.1` resides. When this parameter is defined to a blank space or comma separated list, LSF tries each item in the list to find and load `libkopenafs.so` or `libkopenafs.so.1` to create PAG.

If **LSB_AFS_LIB_DIR** is not defined, or if `libkopenafs.so` or `libkopenafs.so.1` cannot be found at the configured locations, LSF will search in six pre-defined directories:

- /lib
- /lib64
- /usr/lib
- /usr/lib64
- /usr/local/lib
- /usr/local/lib64

The search stops as soon as `libkopenafs.so` or `libkopenafs.so.1` is found and PAG is created.

Default

Not defined.

LSB_API_CONNTIMEOUT

The timeout in seconds that **b*** commands wait to connect to LSF.

Syntax

LSB_API_CONNTIMEOUT=*time_seconds*

Description

The **LSB_API_CONNTIMEOUT** parameter applies to the "handshake" with the **mbatchd** daemon that a **b*** command like **bjobs** first attempts to perform. After the handshake is done, and the command is waiting for the actual contents of the request, the related **LSB_API_RECVTIMEOUT** parameter applies.

Valid values

Any positive integer or zero

Default

10

See also

LSB_API_RECVTIMEOUT

LSB_API_RECVTIMEOUT

Timeout in seconds that **b*** commands wait for a reply from LSF.

Syntax

LSB_API_RECVTIMEOUT=*time_seconds*

Description

After a **b*** command like **bjobs** contacts the **mbatchd** daemon, the command waits for the actual contents of the request. The **LSB_API_RECVTIMEOUT** parameter applies to this time period. The related **LSB_API_CONNTIMEOUT** parameter applies to the "handshake" with the **mbatchd** daemon that **b*** commands first attempt to perform when it connects with **mbatchd**.

Valid values

Any positive integer or zero

Default

10

See also

LSB_API_CONNTIMEOUT

LSB_API_VERBOSE

Syntax

LSB_API_VERBOSE=Y | N

Description

When **LSB_API_VERBOSE=Y**, LSF batch commands will display a retry error message to `stderr` when LIM is not available:

```
LSF daemon (LIM) not responding ... still trying
```

When **LSB_API_VERBOSE=N**, LSF batch commands will not display a retry error message when LIM is not available.

Default

Y. Retry message is displayed to `stderr`.

LSB_BHOSTS_FORMAT

Syntax

LSB_BHOSTS_FORMAT="*field_name*[:[-][*output_width*]] ... [delimiter=*'character'*]"

Description

Sets the customized output format for the **bhosts** command.

- Specify which **bhosts** fields (or aliases instead of the full field names), in which order, and with what width to display.
- Specify only the **bhosts** field name or alias to set its output to unlimited width and left justification.

- Specify the colon (:) without a width to set the output width to the recommended width for that field.
- Specify the colon (:) with a width to set the maximum number of characters to display for the field. When its value exceeds this width, **bhosts** truncates the ending characters.
- Specify a hyphen (-) to set right justification when **bhosts** displays the output for the specific field. If not specified, the default is to set left justification when **bhosts** displays the output for a field.
- Use `delimiter=` to set the delimiting character to display between different headers and fields. This delimiter must be a single character. By default, the delimiter is a space.

Output customization applies only to the output for certain **bhosts** options:

- **LSB_BHOSTS_FORMAT** and **bhosts -o** both apply to output for the **bhosts** command with no options, and for **bhosts** options with output that filter information, including the following options: `-a`, `-alloc`, `-cname`, `-R`, `-x`, `-X`.
- **LSB_BHOSTS_FORMAT** and **bhosts -o** do not apply to output for **bhosts** options that use a modified format, including the following options: `-aff`, `-e`, `-l`, `-s`, `-w`.

The **bhosts -o** option overrides the **LSB_BHOSTS_FORMAT** environment variable, which overrides the **LSB_BHOSTS_FORMAT** setting in `lsf.conf`.

Valid values

The following are the field names used to specify the **bhosts** fields to display, recommended width, aliases you can use instead of field names, and units of measurement for the displayed field:

Field name	Width	Aliases	Unit
host_name	20	hname	
status	15	stat	
cpuf	10		
jl_u	8	jlu	
max	8		
njobs	8		
run	8		
ssusp	8		
ususp	8		
rsv	8		
dispatch_window	50	dispwin	
ngpus	8	ng	
ngpus_alloc	8	ngu	
ngpus_excl_alloc	8	ngx	
ngpus_shared_alloc	8	ngs	
ngpus_shared_jexcl_alloc	8	ngsjx	
ngpus_excl_avail	8	ngfx	
ngpus_shared_avail	8	ngfs	

Field names and aliases are not case-sensitive. Valid values for the output width are any positive integer 1 - 4096.

Example

```
LSB_BHOSTS_FORMAT="host_name cpuf: jl_u:- max:-6 delimiter='^'"
```

Running **bhosts** displays the following fields:

- HOST_NAME with unlimited width and left-aligned.
- CPUF with a maximum width of 10 characters (which is the recommended width) and left-aligned.
- JL_U with a maximum width of 8 characters (which is the recommended width) and right-aligned.
- MAX with a maximum width of 6 characters and right-aligned.
- The ^ character is displayed between different headers and fields.

Default

Not defined. The current **bhosts** output is used.

LSB_BJOBS_CONSISTENT_EXIT_CODE

Syntax

```
LSB_BJOBS_CONSISTENT_EXIT_CODE=Y | N
```

Description

When `LSB_BJOBS_CONSISTENT_EXIT_CODE=Y`, the **bjobs** command exits with 0 only when unfinished jobs are found, and 255 when no jobs are found, or a non-existent job ID is entered.

No jobs are running:

```
bjobs
```

```
No unfinished job found
```

```
echo $?
```

```
255
```

Job 123 does not exist:

```
bjobs 123
```

```
Job <123> is not found
```

```
echo $?
```

```
255
```

Job 111 is running:

```
bjobs 111
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
111	user1	RUN	normal	hostA	hostB	myjob	Oct 22 09:22

```
echo $?
```

```
0
```

Job 111 is running, and job 123 does not exist:

```
bjobs 111 123
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
111	user1	RUN	normal	hostA	hostB	myjob	Oct 22 09:22

```
Job <123> is not found
```

```
echo $?
```

```
255
```

Job 111 is finished:

```
bjobs 111
```

```
No unfinished job found
```

```
echo $?
```

```
255
```

When LSB_BJOBS_CONSISTENT_EXIT_CODE=N, the **bjobs** command exits with 255 only when a non-existent job ID is entered. **bjobs** returns 0 when no jobs are found, all jobs are finished, or if at least one job ID is valid.

No jobs are running:

```
bjobs
```

```
No unfinished job found
```

```
echo $?
```

```
0
```

Job 123 does not exist:

```
bjobs 123
```

```
Job <123> is not found
```

```
echo $?
```

```
0
```

Job 111 is running:

```
bjobs 111
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
111	user1	RUN	normal	hostA	hostB	myjob	Oct 22 09:22

```
echo $?
```

```
0
```

Job 111 is running, and job 123 does not exist:

```
bjobs 111 123
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
111	user1	RUN	normal	hostA	hostB	myjob	Oct 22 09:22

```
Job <123> is not found
```

```
echo $?
```

```
255
```

Job 111 is finished:

```
bjobs 111
```

```
No unfinished job found
```

```
echo $?
```

```
0
```

Default

N.

LSB_BJOBS_FORMAT

Syntax

```
LSB_BJOBS_FORMAT="field_name[:[:-][output_width]] ... [delimiter='character']"
```

Description

Sets the customized output format for the **bjobs** command.

- Specify which **bjobs** fields (or aliases instead of the full field names), in which order, and with what width to display.
- Specify only the **bjobs** field name or alias to set its output to unlimited width and left justification.
- Specify the colon (:) without a width to set the output width to the recommended width for that field.
- Specify the colon (:) with a width to set the maximum number of characters to display for the field. When its value exceeds this width, **bjobs** truncates the output:
 - For the JOB_NAME field, **bjobs** removes the header characters and replaces them with an asterisk (*)
 - For other fields, **bjobs** truncates the ending characters
- Specify a hyphen (-) to set right justification when **bjobs** displays the output for the specific field. If not specified, the default is to set left justification when **bjobs** displays the output for a field.
- Use *delimiter=* to set the delimiting character to display between different headers and fields. This delimiter must be a single character. By default, the delimiter is a space.

Output customization applies only to the output for certain **bjobs** options:

- **LSB_BJOBS_FORMAT** and **bjobs -o** both apply to output for the **bjobs** command with no options, and for **bjobs** options with short form output that filter information, including the following options: -a, -app, -cname, -d, -g, -G, -J, -Jd, -Lp, -m, -P, -q, -r, -sla, -u, -x, -X.
- **LSB_BJOBS_FORMAT** does not apply to output for **bjobs** options that use a modified format and filter information, but you can use **bjobs -o** to customize the output for these options. These options include the following options: -fwd, -N, -p, -s.
- **LSB_BJOBS_FORMAT** and **bjobs -o** do not apply to output for **bjobs** options that use a modified format, including the following options: -A, -aff, -aps, -l, -UF, -ss, -sum, -UF, -w, -W, -WF, -WL, -WP.

The **bjobs -o** option overrides the **LSB_BJOBS_FORMAT** environment variable, which overrides the **LSB_BJOBS_FORMAT** setting in `lsf.conf`.

Valid values

The following are the field names used to specify the **bjobs** fields to display, recommended width, aliases you can use instead of field names, and units of measurement for the displayed field:

Table 6. Output fields for bjobs

Field name	Width	Aliases	Unit	Category
jobid	7	id		Common
jobindex	8			
stat	5			
user	7			
user_group	15	ugroup		
queue	10			
job_name	10	name		
job_description	17	description		
proj_name	11	proj, project		
application	13	app		
service_class	13	sla		
job_group	10	group		
job_priority	12	priority		
rsvid	40			
esub	20			
kill_reason	50			
dependency	15			
pend_reason	11			
charged_saap	50	saap		
command	15	cmd		
pre_exec_command	16	pre_cmd		
post_exec_command	17	post_cmd		
resize_notification_command	27	resize_cmd		
pids	20			
exit_code	10			
exit_reason	50			
interactive	11			

Common

Table 6. Output fields for bjobs (continued)

Field name	Width	Aliases	Unit	Category
from_host	11			Host
first_host	11			
exec_host	11			
nexec_host Note: If the allocated host group or compute unit is condensed, this field does not display the real number of hosts. Use bjobs -X -o to view the real number of hosts in these situations.	10			
ask_hosts	30			
alloc_slot	20			
nalloc_slot	10			
host_file	10			
exclusive	13			
nreq_slot	10			

Table 6. Output fields for bjobs (continued)

Field name	Width	Aliases	Unit	Category
submit_time	15		time stamp	Time
start_time	15		time stamp	
estimated_start_time	20	estart_time	time stamp	
specified_start_time	20	sstart_time	time stamp	
specified_terminate_time	24	sterminate_time	time stamp	
time_left	11		seconds	
finish_time	16		time stamp	
estimated_run_time	20	ertime	seconds	
ru_utime	12		seconds	
ru_stime	12		seconds	
%complete	11			
warning_action	15	warn_act		
action_warning_time	19	warn_time		
pendstate (IPEND/EPEND/ NOTPEND)	9			
pend_time	12		seconds	
ependtime	12		seconds	
ipendtime	12		seconds	
estimated_sim_start_time	24	esstart_time	time stamp	
effective_plimit (run with bjobs -p to show information for pending jobs only)	18		seconds	
plimit_remain (run with bjobs -p to show information for pending jobs only) A negative number indicates the amount of time in which the job exceeded the pending time limit, while a positive number shows that the time remaining until the pending time limit is reached.	15		seconds	
effective_eplimit (run with bjobs -p to show information for pending jobs only)	19		seconds	
eplimit_remain (run with bjobs -p to show information for pending jobs only)	16		seconds	

Field name	Width	Aliases	Unit	Category
cpu_used	10			CPU
run_time	15		seconds	
idle_factor	11			
exception_status	16	except_stat		
slots	5			
mem	10		LSF_UNIT_FOR_LIMIT S in lsf.conf (KB by default)	
max_mem	10		LSF_UNIT_FOR_LIMIT S in lsf.conf (KB by default)	
avg_mem	10		LSF_UNIT_FOR_LIMIT S in lsf.conf (KB by default)	
memlimit	10		LSF_UNIT_FOR_LIMIT S in lsf.conf (KB by default)	
swap	10		LSF_UNIT_FOR_LIMIT S in lsf.conf (KB by default)	
swplimit	10		LSF_UNIT_FOR_LIMIT S in lsf.conf (KB by default)	
gpu_num	10	gnum		GPU
gpu_mode	20	gmode		
j_exclusive	15	j_excl		
gpu_alloc	30	galloc		
nthreads	10			Resource usage
hrusage	50			
min_req_proc	12			Resource requirement
max_req_proc	12			
effective_resreq	17	eresreq		
combined_resreq	20	cresreq		
network_req	15			

Field name	Width	Aliases	Unit	Category
filelimit	10			Resource limits
corelimit	10			
stacklimit	10			
processlimit	12			
runtimelimit	12			
plimit	10		seconds	
eplimit	10		seconds	
input_file	10			File
output_file	11			
error_file	10			
output_dir	15			Directory
sub_cwd	10			
exec_home	10			
exec_cwd	10			
licproject	20			
forward_cluster	15	fwd_cluster		MultiCluster
forward_time	15	fwd_time	time stamp	
srcjobid	8			
dstjobid	8			
source_cluster	15	srcluster		
energy			Joule	
gpfsio				Energy
Job disk usage (I/O) data on IBM Spectrum Scale.				

Field names and aliases are not case-sensitive. Valid values for the output width are any positive integer 1 - 4096. If the jobid field is defined with no output width and **LSB_JOBID_DISP_LENGTH** is defined in `lsf.conf`, the **LSB_JOBID_DISP_LENGTH** value is used for the output width. If jobid is defined with a specified output width, the specified output width overrides the **LSB_JOBID_DISP_LENGTH** value.

Example

```
LSB_BJOBS_FORMAT="jobid stat: queue:- project:10 application:-6 delimiter='^'"
```

Running **bjobs** displays the following fields:

- JOBID with unlimited width and left-aligned. If **LSB_JOBID_DISP_LENGTH** is specified, that value is used for the output width instead.
- STAT with a maximum width of 5 characters (which is the recommended width) and left-aligned.
- QUEUE with a maximum width of 10 characters (which is the recommended width) and right-aligned.
- PROJECT with a maximum width of 10 characters and left-aligned.

- APPLICATION with a maximum width of 6 characters and right-aligned.
- The ^ character is displayed between different headers and fields.

Default

Not defined. The current **bjobs** output is used.

LSB_BJOBS_PENDREASON_LEVEL**Syntax**

LSB_BJOBS_PENDREASON_LEVEL=0|1|2|3

Description

Sets the default behavior for displaying pending reasons when a user enters the **bjobs -p** or **bsub -I** command. For **bjobs -p**, the default is used if the user does not specify a level of 0 to 3 with the **-p** option.

Values

- 0: Displays pending reason as before the LSF 10.1 release.
- 1: Displays the single key pending reason.
- 2: Displays categorized, host-based pending reasons for candidate hosts in the cluster. For the candidate hosts, the actual reason on each host is shown. For each pending reason, the number of hosts that give the reason is shown. The actual pending reason messages appear from most to least common.
- 3: Displays categorized, host-based pending reasons for both candidate and non-candidate hosts in the cluster. For both the candidate and non-candidate hosts, the actual pending reason on each host is shown. For each pending reason, the number of hosts that show that reason is given. The actual reason messages appear from most to least common.

Default

Set to 0 by default.

LSB_BLOCK_JOBINFO_TIMEOUT**Syntax**

LSB_BLOCK_JOBINFO_TIMEOUT=*time_minutes*

Description

Timeout in minutes for job information query commands (e.g., **bjobs**).

Valid values

Any positive integer

Default

Not defined (no timeout)

See also

MAX_JOBINFO_QUERY_PERIOD in `lsb.params`

LSB_BMGROUP_ALLREMOTE_EXPAND

This `lsf.conf` configuration option selects the display of leased-in hosts as a list, or replaces the list with the `allremote` keyword in `bmgroup` output.

Syntax

LSB_BMGROUP_ALLREMOTE_EXPAND=N | n

Description

This `lsf.conf` configuration option can be defined to `Y|y` or `N|n`. If the value is `Y|y`, leased-in hosts are displayed as a list of hosts in the form `host_name@cluster_name` (for example, `hosta@cluster1`).

If the value is `N|n`, leased-in hosts are represented by a single keyword `allremote` instead of being displayed as a list.

Can also be defined as an environment variable.

Default

Y.

LSB_BPEEK_REMOTE_OUTPUT

Syntax

LSB_BPEEK_REMOTE_OUTPUT=y|Y|n|N

Description

If disabled (set to `N`), the `bpeek` command attempts to retrieve the job output from the local host first. If that fails, `bpeek` attempts to retrieve the job output from the remote host instead.

If enabled (set to `Y`), it is the opposite. The `bpeek` command attempts to retrieve the job output from the remote host first, then the local host.

When attempting to retrieve the job output from the remote host, `bpeek` attempts to use `RES` first, then `rsh`. If neither is running on the remote host, the `bpeek` command cannot retrieve job output.

Best Practices

Three directories are related to the `bpeek` command:

- the user's home directory
- the `JOB_SPOOL_DIR` directory
- the checkpoint directory

If these directories are on a shared file system, this parameter can be disabled.

If any of these directories are not on a shared file system, this parameter should be enabled, and either `RES` or `rsh` should be started on the remote job execution host.

Default

N

LSB_BSUB_ERR_RETRY

Syntax

LSB_BSUB_ERR_RETRY=RETRY_CNT [*integer*] ERR_TYPE [*error1* [*error2*] [...]]

Description

In some cases, jobs can benefit from being automatically retried in the case of failing for a particular error. When specified, **LSB_BSUB_ERR_RETRY** automatically retries jobs that exit with a particular reason, up to the number of times specified by **RETRY_CNT**.

Only the following error types (**ERR_TYPE**) are supported:

- **BAD_XDR**: Error during XDR.
- **MSG_SYS**: Failed to send or receive a message.
- **INTERNAL**: Internal library error.

The number of retries (**RETRY_CNT**) can be a minimum of 1 to a maximum of 50.

Considerations when setting this parameter:

- Users may experience what seems like a lag during job submission while the job is retried automatically in the background.
- Users may see a job submitted more than once, with no explanation (no error is communicated to the user; the job keeps getting submitted until it succeeds or reaches its maximum retry count). In this case, the job ID also changes each time the error is retried.

Default

Not defined. If retry count is not valid, defaults to 5.

LSB_BSUB_PARSE_SCRIPT

Specifies whether you can use the **bsub** command to parse and run job scripts from the command line.

Syntax

LSB_BSUB_PARSE_SCRIPT=Y|y|N|n

Description

When set to Y|y, you can use the **bsub** command to open, parse, and run job scripts directly from the command line. Submit a job with the job script as a command. The job script must be an ASCII text file and not a binary file.

By default (or when set to N|n), the **bsub** command does not open, parse, and run job scripts directly from the command line. **bsub** runs them as regular executable jobs.

- Write a job file one line at a time. Run **bsub** without specifying a command, then specify valid Bourne shell commands or Windows batch file command lines at each **bsub** > prompt.
- Use the < redirect to specify an ASCII text file that contains either Bourne shell command lines or Windows batch file command lines.

Use the **#BSUB** imperative (in upper case letters) at the beginning of each line to specify job submission options in the script.

Example

For example, the following script (**myscript.sh**) uses the **#BSUB** imperative to run the **myjob1 arg1** and **myjob2 arg2** commands with the **bsub -n 2** and **-P myproj** options:

```
#!/bin/sh
#BSUB -n 2
myjob1 arg1
myjob2 arg2
#BSUB -P myproj
```

By default, run the following command to use this job script:

```
bsub myscript.sh
```

If `LSB_BSUB_PARSE_SCRIPT=N` is set, run the following command to use this job script:

```
bsub < myscript.sh
```

Alternatively, manually specify job submission options one line at a time to run the same commands as the previous `myscript.sh` file:

```
% bsub
bsub> #BSUB -n 2
bsub> myjob1 arg1
bsub> myjob2 arg2
bsub> #BSUB -P myproj
bsub> ^D
```

Default

N

Related concepts

[Write job scripts](#)

LSB_BUSERS_FORMAT

Syntax

```
LSB_BUSERS_FORMAT="field_name ..."
```

Description

Sets the customized output format for the **busers** command. Specify which **busers** fields to display, and in which order.

The **busers -o** option overrides the **LSB_BUSERS_FORMAT** environment variable, which overrides the **LSB_BUSERS_FORMAT** setting in `lsf.conf`.

Valid values

The following are the field names used to specify the **busers** fields to display:

- user (the name of the user or user group)
- jl/p
- max
- nstart (the current number of starting tasks for all of a users' jobs)
- pend
- run
- ssusp
- ususp
- rsv
- njobs
- pjobs
- mpend
- mpjobs
- priority

lsf.conf

- `ngpus` (the number of physical GPUs that the users or user groups are using)
- `ngpus_shared` (the number of physical GPUs that the users or user groups are using in shared mode)
- `ngpus_excl` (the number of physical GPUs that the users or user groups are using in exclusive mode)
- `ngpus_shared_jexcl` (the number of physical GPUs that the users or user groups are using in shared mode, but the jobs of the user is exclusive)

Field names are case-sensitive.

Example

```
LSB_BUSERS_FORMAT="user ngpus_alloc ngpus_excl_alloc ngpus_shared_alloc  
ngpus_shared_jexcl_alloc"
```

Default

Not defined. The current **busers** output is used.

LSB_BWAIT_REREG_INTERVAL

Syntax

```
LSB_BWAIT_REREG_INTERVAL=minutes
```

Description

Specifies the default time interval to reregister the wait condition from the **bwait** command to the **mbatchd** daemon, in minutes.

If you do not run the **bwait -r** option, the **bwait** command uses this parameter value as the default.

Valid values

An integer between 1 and 525600 (one year).

Default

10 minutes.

LSB_CALL_LIM_RETRY

Specifies the maximum number of times that the **mbatchd** daemon attempts to contact the master LIM when **mbatchd** cannot detect the LIM.

Syntax

```
LSB_CALL_LIM_RETRY=integer
```

Description

The **mbatchd** daemon periodically checks the master LIM to make sure that it is up. If the LIM is temporarily down (for example, after restarting or reconfiguring a cluster), this parameter specifies the maximum number of times that **mbatchd** attempts to contact the LIM before it exits. **mbatchd** waits for 2 seconds before attempting to contact the LIM again.

For medium clusters (100 - 1000 hosts), specify 40. Increase this number for larger clusters.

Valid values

Any positive integer or zero

Default

0.

LSB_CGROUP_MEM_INCLUDE_CACHE

Specifies whether LSF includes rss and cache in the memory usage report when Linux cgroups are enabled.

Syntax

```
LSB_CGROUP_MEM_INCLUDE_CACHE=Y|y|N|n
```

Description

When set to Y|y, LSF includes rss and cache in the memory usage report when Linux cgroups are enabled.

When set to N|n, LSF only includes rss in the memory usage report when Linux cgroups are enabled.

Short jobs might not have memory usage to be displayed when the **LSB_CGROUP_MEM_INCLUDE_CACHE** is not defined or defined as N.

Default

N

LSB_CHECK_JOB_PID_REUSE_ON_REBOOT**Syntax**

```
LSB_CHECK_JOB_PID_REUSE_ON_REBOOT=y|Y|n|N
```

Description

If set, LSF will create a `jobid.processid.stt` file under `LSF_TMPDIR/.clustername.sbd/` on the execution host when a job is started, and will clean it when job is finished. When the execution host is rebooted, LSF will check if the job process id has been reused by another process. If it has, the job status will change to exit status.

Default

n|N. Do not check if the host job pid is reused when the host is rebooted.

LSB_CHUNK_RUSAGE**Syntax**

```
LSB_CHUNK_RUSAGE=y
```

Description

Applies only to chunk jobs. When set, **sbatchd** contacts PIM to retrieve resource usage information to enforce resource usage limits on chunk jobs.

By default, resource usage limits are not enforced for chunk jobs because chunk jobs are typically too short to allow LSF to collect resource usage.

If `LSB_CHUNK_RUSAGE=Y` is defined, limits may not be enforced for chunk jobs that take less than a minute to run.

Default

Not defined. No resource usage is collected for chunk jobs.

LSB_CMD_LOG_MASK**Syntax**

LSB_CMD_LOG_MASK=*log_level*

Description

Specifies the logging level of error messages from LSF batch commands.

To specify the logging level of error messages for LSF commands, use LSB_CMD_LOG_MASK. To specify the logging level of error messages for LSF daemons, use LSF_LOG_MASK.

LSB_CMD_LOG_MASK sets the log level and is used in combination with LSB_DEBUG_CMD, which sets the log class for LSF batch commands. For example:

```
LSB_CMD_LOG_MASK=LOG_DEBUG LSB_DEBUG_CMD="LC_TRACE LC_EXEC"
```

LSF commands log error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by LSB_CMD_LOG_MASK determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used for basic debugging. The level LOG_DEBUG3 records all debugging messages, and can cause log files to grow very large; it is not often used. Most debugging is done at the level LOG_DEBUG2.

The commands log to the syslog facility unless LSB_CMD_LOGDIR is set.

Valid values

The log levels from highest to lowest are:

- LOG_EMERG
- LOG_ALERT
- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

Default

LOG_WARNING

See also

LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSB_CMD_LOGDIR

Syntax

LSB_CMD_LOGDIR=*path*

Description

Specifies the path to the LSF command log files.

Default

/tmp

See also

LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSB_CONFDIR

Syntax

LSB_CONFDIR=*path*

Description

Specifies the path to the directory containing the LSF configuration files.

The configuration directories are installed under **LSB_CONFDIR**.

Configuration files for each cluster are stored in a subdirectory of **LSB_CONFDIR**. This subdirectory contains several files that define user and host lists, operation parameters, and queues.

All files and directories under **LSB_CONFDIR** must be readable from all hosts in the cluster. LSB_CONFDIR/*cluster_name*/*configdir* must be owned by the LSF administrator.

If live reconfiguration through the **bconf** command is enabled by the parameter **LSF_LIVE_CONFDIR**, configuration files are written to and read from the directory set by **LSF_LIVE_CONFDIR**.

Do not change this parameter after LSF has been installed.

Default

LSF_CONFDIR/lsbatch

See also

LSF_CONFDIR, LSF_LIVE_CONFDIR

LSB_CPUSET_BESTCPUS

Syntax

LSB_CPUSET_BESTCPUS=*y* | *Y*

Description

If set, enables the best-fit algorithm for SGI cpusets

Default

Y (best-fit)

LSB_CPUSET_DISPLAY_CPULIST

Syntax

LSB_CPUSET_DISPLAY_CPULIST=Y | N

Description

The **bjobs/bhist/bacct -l** commands display the CPU IDs in the dynamic CPUset allocated on each host. The CPU IDs are displayed as CPUS=cpu_ID_list after NCPUS=num_cpus for each host. The cpu_ID_list is displayed in condensed format as a range of continuous IDs.

After enabling **LSB_CPUSET_DISPLAY_CPULIST** in lsf.conf, the LSF administrator must run **badmin reconfig** to make the change effective. CPU IDs are shown in **bjobs/bhist/bacct -l** for the cpuset jobs dispatched after **badmin reconfig**.

Default

N. **bjobs/bhist/bacct -l** do not display CPU IDs for cpusets allocated on each host.

LSB_DEBUG

Syntax

LSB_DEBUG=1 | 2

Description

Sets the LSF batch system to debug.

If defined, LSF runs in single user mode:

- No security checking is performed
- Daemons do not run as root

When LSB_DEBUG is defined, LSF does not look in the system services database for port numbers. Instead, it uses the port numbers defined by the parameters LSB_MBD_PORT/LSB_SBD_PORT in lsf.conf. If these parameters are not defined, it uses port number 40000 for mbatchd and port number 40001 for sbatchd.

You should always specify 1 for this parameter unless you are testing LSF.

Can also be defined from the command line.

Valid values

LSB_DEBUG=1

The LSF system runs in the background with no associated control terminal.

LSB_DEBUG=2

The LSF system runs in the foreground and prints error messages to tty.

Default

Not defined

See also

LSB_DEBUG, LSB_DEBUG_CMD, LSB_DEBUG_MBD, LSB_DEBUG_NQS, LSB_DEBUG_SBD, LSB_DEBUG_SCH, LSF_DEBUG_LIM, LSF_DEBUG_RES, LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT, LSF_LOGDIR, LSF_LIM_DEBUG, LSF_RES_DEBUG

LSB_DEBUG_CMD**Syntax**

```
LSB_DEBUG_CMD=log_class
```

Description

Sets the debugging log class for commands and APIs.

Specifies the log class filtering to be applied to LSF batch commands or the API. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_CMD sets the log class and is used in combination with LSB_CMD_LOG_MASK, which sets the log level. For example:

```
LSB_CMD_LOG_MASK=LOG_DEBUG LSB_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Debugging is turned on when you define both parameters.

The daemons log to the sysLog facility unless LSB_CMD_LOGDIR is defined.

To specify multiple log classes, use a space-separated list enclosed by quotation marks. For example:

```
LSB_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Can also be defined from the command line.

Valid values

Valid log classes are:

- LC_ADVRSV and LC2_ADVRSV: Log advance reservation modifications
- LC_AFS and LC2_AFS: Log AFS messages
- LC_AUTH and LC2_AUTH: Log authentication messages
- LC_CHKPNT and LC2_CHKPNT: Log checkpointing messages
- LC_COMM and LC2_COMM: Log communication messages
- LC_DCE and LC2_DCE: Log messages pertaining to DCE support
- LC_EEVENTD and LC2_EEVENTD: Log eeventd messages
- LC_ELIM and LC2_ELIM: Log ELIM messages
- LC_EXEC and LC2_EXEC: Log significant steps for job execution
- LC_FAIR and LC2_FAIR: Log fairshare policy messages
- LC_FILE and LC2_FILE: Log file transfer messages
- LC_FLEX and LC2_FLEX: Log messages related to FlexNet
- LC2_GUARANTEE: Log messages related to guarantee SLAs
- LC_HANG and LC2_HANG: Mark where a program might hang
- LC_JARRAY and LC2_JARRAY: Log job array messages
- LC_JLIMIT and LC2_JLIMIT: Log job slot limit messages
- LC2_LIVECONF: Log live reconfiguration messages
- LC_LOADINDX and LC2_LOADINDX: Log load index messages

- LC_M_LOG and LC2_M_LOG: Log multievent logging messages
- LC_MEMORY and LC2_MEMORY: Log messages related to MEMORY allocation
- LC_MPI and LC2_MPI: Log MPI messages
- LC_MULTI and LC2_MULTI: Log messages pertaining to MultiCluster
- LC_PEND and LC2_PEND: Log messages related to job pending reasons
- LC_PERFM and LC2_PERFM: Log performance messages
- LC_PIM and LC2_PIM: Log PIM messages
- LC_PREEMPT and LC2_PREEMPT: Log preemption policy messages
- LC_RESOURCE and LC2_RESOURCE: Log messages related to resource broker
- LC_RESREQ and LC2_RESREQ: Log resource requirement messages
- LC_SCHED and LC2_SCHED: Log messages pertaining to the mbatchd scheduler.
- LC_SIGNAL and LC2_SIGNAL: Log messages pertaining to signals
- LC_SYS and LC2_SYS: Log system call messages
- LC_TRACE and LC2_TRACE: Log significant program walk steps
- LC_XDR and LC2_XDR: Log everything transferred by XDR
- LC_XDRVERSION and LC2_XDRVERSION: Log messages for XDR version
- LC2_KRB: Log message related to Kerberos integration
- LC2_DC: Log message related to Dynamic Cluster
- LC2_CGROUP: Log message related to cgroup operation
- LC2_TOPOLOGY: Log message related to hardware topology
- LC2_AFFINITY: Log message related to affinity
- LC2_LSF_PE: Log message related to LSF PE integration
- LC2_DAS: Log message related to LSF data manager
- LC2_RC: Log resource connector messages

Default

Not defined

See also

LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_MBD, LSB_DEBUG_NQS, LSB_DEBUG_SBD, LSB_DEBUG_SCH, LSF_DEBUG_LIM, LSF_DEBUG_RES, LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT, LSF_LOGDIR, LSF_LIM_DEBUG, LSF_RES_DEBUG

LSB_DEBUG_GPD**Syntax**

LSB_DEBUG_GPD=log_class

Description

Sets the debugging log class for **gpolicyd**. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_GPD is used in combination with **LSF_LOG_MASK** which sets the log level. For example:

- **LSF_LOG_MASK**=LOG_DEBUG
- **LSB_DEBUG_GPD**="LC_TRACE LC_COMM"

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_GPD="LC_TRACE LC2_G_FAIR"
```

You must restart the daemons after setting **LSB_DEBUG_GPD** for your changes to take effect.

Valid values

- LC_AUTH - Log authentication messages
- LC_COMM - Log communication messages
- LC_SYS - Log system call messages
- LC_TRACE - Log significant program walk steps
- LC_XDR - Log everything transferred by XDR
- LC_XDRVERSION - Log messages for XDR version
- LC2_G_FAIR - Log messages for global fairshare.

Default

Not defined.

LSB_DEBUG_MBD

Syntax

```
LSB_DEBUG_MBD=log_class
```

Description

Sets the debugging log class for mbatchd.

Specifies the log class filtering to be applied to mbatchd. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_MBD sets the log class and is used in combination with **LSF_LOG_MASK**, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG LSB_DEBUG_MBD="LC_TRACE LC_EXEC"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_MBD="LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting **LSB_DEBUG_MBD** for your changes to take effect.

If you use the command **badmin mbddebug** to temporarily change this parameter without changing `lsf.conf`, you do not need to restart the daemons.

Valid values

Valid log classes are the same as for **LSB_DEBUG_CMD** except for the log class **LC_ELIM**, which cannot be used with **LSB_DEBUG_MBD**. See **LSB_DEBUG_CMD**.

Default

Not defined

See also

LSB_CMD_LOG_MASK, **LSB_CMD_LOGDIR**, **LSB_DEBUG**, **LSB_DEBUG_MBD**, **LSB_DEBUG_NQS**, **LSB_DEBUG_SBD**, **LSB_DEBUG_SCH**, **LSF_DEBUG_LIM**, **LSF_DEBUG_RES**, **LSF_LIM_PORT**, **LSF_RES_PORT**, **LSB_MBD_PORT**, **LSB_SBD_PORT**, **LSF_LOGDIR**, **LSF_LIM_DEBUG**, **LSF_RES_DEBUG**

LSB_DEBUG_SBD

Syntax

LSB_DEBUG_SBD=*log_class*

Description

Sets the debugging log class for sbatchd.

Specifies the log class filtering to be applied to sbatchd. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_SBD sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG LSB_DEBUG_SBD="LC_TRACE LC_EXEC"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_SBD="LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSB_DEBUG_SBD for your changes to take effect.

If you use the command **badmin sbddebug** to temporarily change this parameter without changing lsf.conf, you do not need to restart the daemons.

Valid values

Valid log classes are the same as for **LSB_DEBUG_CMD** except for the log class **LC_ELIM**, which cannot be used with **LSB_DEBUG_SBD**. See **LSB_DEBUG_CMD**.

Default

Not defined

See also

LSB_DEBUG_MBD, **LSF_CMD_LOGDIR**, **LSF_CMD_LOG_MASK**, **LSF_LOG_MASK**, **LSF_LOGDIR**, **badmin**

LSB_DEBUG_SCH

Syntax

LSB_DEBUG_SCH=*log_class*

Description

Sets the debugging log class for mbschd.

Specifies the log class filtering to be applied to mbschd. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_SCH sets the log class and is used in combination with **LSF_LOG_MASK**, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG LSB_DEBUG_SCH="LC_SCHED"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_SCH="LC_SCHED LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting **LSB_DEBUG_SCH** for your changes to take effect.

Valid values

Valid log classes are the same as for **LSB_DEBUG_CMD** except for the log class **LC_ELIM**, which cannot be used with **LSB_DEBUG_SCH**, and **LC_HPC** and **LC_SCHED**, which are only valid for **LSB_DEBUG_SCH**. See **LSB_DEBUG_CMD**.

Default

Not defined

See also

LSB_DEBUG_MBD, **LSB_DEBUG_SBD**, **LSF_CMD_LOGDIR**, **LSF_CMD_LOG_MASK**, **LSF_LOG_MASK**, **LSF_LOGDIR**, **badadmin**

LSB_DISABLE_LIMLOCK_EXCL**Syntax**

LSB_DISABLE_LIMLOCK_EXCL=y | n

Description

If preemptive scheduling is enabled, this parameter enables preemption of and preemption by exclusive jobs when **PREEMPT_JOBTYPE=EXCLUSIVE** in **lsb.params**. Changing this parameter requires a restart of all sbatchds in the cluster (**badadmin hrestart**). Do not change this parameter while exclusive jobs are running.

When **LSB_DISABLE_LIMLOCK_EXCL=y**, for a host running an exclusive job:

- LIM is not locked on a host running an exclusive job
- **lsload** displays the host status ok.
- **bhosts** displays the host status closed.
- Users can run tasks on the host using **lsrun** or **lsgrun**. To prevent users from running tasks during execution of an exclusive job, the parameter **LSF_DISABLE_LSRUN=y** must be defined in **lsf.conf**.

Default

Set to y at time of installation. If otherwise undefined, then n (LSF locks the LIM on a host running an exclusive job and unlocks the LIM when the exclusive job finishes).

LSB_DISABLE_RERUN_POST_EXEC**Syntax**

LSB_DISABLE_RERUN_POST_EXEC=y | Y

Description

If set, and the job is rerunnable, the **POST_EXEC** configured at the job level or the queue level is not executed if the job is rerun.

Running of post-execution commands upon restart of a rerunnable job may not always be desirable. For example, if the post-exec removes certain files, or does other cleanup that should only happen if the job finishes successfully, use **LSB_DISABLE_RERUN_POST_EXEC** to prevent the post-exec from running and allow the successful continuation of the job when it reruns.

The **POST_EXEC** may still run for a job rerun when the execution host loses contact with the master host due to network problems. In this case mbatchd assumes the job has failed and restarts the job on another host. The original execution host, out of contact with the master host, completes the job and runs the **POST_EXEC**.

Default

Not defined

LSB_DISABLE_SUSPEND_AR_JOBS

Syntax

LSB_DISABLE_SUSPEND_AR_JOBS=y | Y

Description

This parameter controls the behavior of non-AR jobs whose reservation has been deleted. If the parameter, LSB_DISABLE_SUSPEND_AR_JOBS is set to N, those jobs whose reservation has been deleted will be suspended. If there are any available slots, these jobs will run after the temporary suspension; however, if the cluster has other higher priority jobs, then the regular job will be kept as suspended.

If the parameter, LSB_DISABLE_SUSPEND_AR_JOBS is set to Y, those jobs whose reservation has been deleted will be kept running. Any jobs that were suspended when the AR started will continue to be suspended, even if they are higher priority than the AR jobs.

Default

LSB_DISABLE_SUSPEND_AR_JOBS=n | N

LSB_DISPATCH_CHECK_RESUME_INTVL

Syntax

LSB_DISPATCH_CHECK_RESUME_INTVL=y|Y|n|N

Description

When this parameter is enabled, LSF takes the last resume time of a host into account when considering dispatching PEND jobs to this host. LSF will not dispatch PEND jobs to a host if it has just resumed a job on this host within the *mbdSleepTime* interval.

After configuring this parameter, run **badmin reconfig** to have it take effect.

Default

N

LSB_DISPLAY_YEAR

Syntax

LSB_DISPLAY_YEAR=y|Y|n|N

Description

Toggles on and off inclusion of the year in the time string displayed by the commands **bjobs -l**, **bacct -l**, and **bhist -l|-b|-t**.

Default

N

LSB_EAUTH_DATA_REUSE

Syntax

LSB_EAUTH_DATA_REUSE=Y | N

Description

When set to Y, **blaunch** caches authentication data when connecting to the first remote execution server in memory. **blaunch** uses this cached data to authenticate subsequent first remote execution servers. If set to N, **blaunch** does not cache authentication data. Every time **blaunch** connects to a different authentication, it fetches new authentication data.

Default

Y

LSB_EAUTH_EACH_SUBPACK

Syntax

LSB_EAUTH_EACH_SUBPACK=Y | N

Description

Enable this parameter to have **bsub** call `eauth` for each sub-pack. **LSB_MAX_PACK_JOBS** defines the number of jobs that one sub-pack contains. If **LSB_EAUTH_EACH_SUBPACK** is not enabled, **bsub** only calls `eauth` for the first sub-pack and caches this `eauth` data. The cached data is reused for each subsequent sub-pack instead of **bsub** calling `eauth` again.

Default

N

LSB_ECHKPNT_KEEP_OUTPUT

Syntax

LSB_ECHKPNT_KEEP_OUTPUT=y | Y

Description

Saves the standard output and standard error of custom **echkpnt** and **erestart** methods to:

- `checkpoint_dir/$LSB_JOBID/echkpnt.out`
- `checkpoint_dir/$LSB_JOBID/echkpnt.err`
- `checkpoint_dir/$LSB_JOBID/erestart.out`
- `checkpoint_dir/$LSB_JOBID/erestart.err`

Can also be defined as an environment variable.

Default

Not defined. Standard error and standard output messages from custom **echkpnt** and **erestart** programs is directed to `/dev/null` and discarded by LSF.

See also

LSB_ECHKPNT_METHOD, **LSB_ECHKPNT_METHOD_DIR**

LSB_ECHKPNT_METHOD

Syntax

LSB_ECHKPNT_METHOD="*method_name* [*method_name*] ..."

Description

Name of custom **echknt** and **erestart** methods.

Can also be defined as an environment variable, or specified through the **bsub -k** option.

The name you specify here is used for both your custom **echknt** and **erestart** programs. You must assign your custom **echknt** and **erestart** programs the name `echknt.method_name` and `erestart.method_name`. The programs `echknt.method_name` and `erestart.method_name` must be in LSF_SERVERDIR or in the directory specified by LSB_ECHKPNT_METHOD_DIR.

Do not define LSB_ECHKPNT_METHOD=default as default is a reserved keyword to indicate to use the default **echknt** and **erestart** methods of LSF. You can however, specify **bsub -k "my_dir method=default" my_job** to indicate that you want to use the default checkpoint and restart methods.

When this parameter is defined, LSF uses the custom checkpoint and restart methods specified.

Limitations

The method name and directory (LSB_ECHKPNT_METHOD_DIR) combination must be unique in the cluster.

For example, you may have two **echknt** applications with the same name such as `echknt.mymethod` but what differentiates them is the different directories defined with LSB_ECHKPNT_METHOD_DIR. It is the cluster administrator's responsibility to ensure that method name and method directory combinations are unique in the cluster.

Default

Not defined.

See also

LSB_ECHKPNT_METHOD_DIR, LSB_ECHKPNT_KEEP_OUTPUT

LSB_ECHKPNT_METHOD_DIR

Syntax

LSB_ECHKPNT_METHOD_DIR=*path*

Description

Absolute path name of the directory in which custom **echknt** and **erestart** programs are located.

The checkpoint method directory should be accessible by all users who need to run the custom **echknt** and **erestart** programs.

Can also be defined as an environment variable.

Default

Not defined. LSF searches in LSF_SERVERDIR for custom **echknt** and **erestart** programs.

See also

LSB_ESUB_METHOD, LSB_ECHKPNT_KEEP_OUTPUT

LSB_ENABLE_ESTIMATION**Syntax**

LSB_ENABLE_ESTIMATION=Y|N

Description

Enables job start time estimation. When set to Y, **mbatchd** uses a simulation-based estimator to predict the start time of jobs.

To view the estimated start time of a job, run **bjobs -l** and refer to the ESTIMATION section, or run **bjobs -o** and specify `estimated_sim_start_time` or the `esstart_time` alias as a display field (for example, run **bjobs -o "esstart_time"**). If **mbatchd** did not estimate the start time for the specified job, **bjobs -l** does not display the ESTIMATION section.

To view the status of the LSF estimator, run **badmin showstatus**.

Default

N.

See also

ESTIMATOR_SIM_START_INTERVAL, **ESTIMATOR_MAX_TIME_PREDICTION**, **ESTIMATOR_MAX_JOBS_PREDICTION**, **ESTIMATOR_MAX_RUNTIME_PREDICTION**, and **ESTIMATOR_CPUS** parameters in `lsb.params`.

LSB_ENABLE_HPC_ALLOCATION**Syntax**

LSB_ENABLE_HPC_ALLOCATION=Y|y|N|n

Description

When set to Y|y, this parameter changes concept of the required number of slots for a job to the required number of tasks for a job. The specified numbers of tasks are the number of tasks to launch on execution hosts. The allocated slots change to all slots on the allocated execution hosts for an exclusive job in order to reflect the actual slot allocation. This parameter improves job scheduling, job accounting, and resource utilization.

When **LSB_ENABLE_HPC_ALLOCATION** is not set or is set to N|n, the following behavior changes still take effect:

- Pending reasons in **bjobs** output keep the task concept.
- Changes for **PROCLIMIT**, **PER_SLOT**, and **IMPT_SLOTBKLK**
- Event and API changes for task concept
- Field "alloc_slot nalloc_slot" for **bjobs -o**
- The `-alloc` option in **bqueues**, **bhosts**, **busers**, and **bapp** remains.
- Command help messages change to the task concept.
- Error messages in logs change to the task concept.

The following behavior changes do NOT take effect:

- Command output for **bjobs**, **bhist**, and **bacct**

- Exclusive job slot allocation change

Important:

- After you change the **LSB_ENABLE_HPC_ALLOCATION** setting, you must run **badm** **mbdrestart** to restart **mbatchd** and make it take effect in **mbatchd**.
- Changing **LSB_ENABLE_HPC_ALLOCATION** takes effect immediately for the affected commands (that is, **bjobs**, **bhist**, **bacct**, **bapp**, **bhosts**, **bmod**, **bqueues**, **bsub**, and **busers**). Therefore, it is not required to restart **mbatchd** to enable the change for **b*** commands.
- For a non-shared LSF installation, **b*** commands take effect on slave hosts according to the local `lsf.conf` setting. If **LSB_ENABLE_HPC_ALLOCATION** is not defined in the local `lsf.conf`, slave **b*** commands call the master LIM and take the configuration from the master host. To allow a server to take the latest configuration from the master host, the master LIM must perform a reconfiguration after configuration changes on the master host.

Default

N

For new installations of LSF, **LSB_ENABLE_HPC_ALLOCATION** is set to Y automatically.

LSB_ENABLE_PERF_METRICS_LOG**Syntax**

LSB_ENABLE_PERF_METRICS_LOG=Y | N

Description

Enable this parameter to have LSF log `mbatchd` performance metrics. In any sample period, data that is not likely to cause performance issues will not be logged. The performance metric data is logged periodically according to the time interval set in **LSB_PERF_METRICS_SAMPLE_PERIOD**.

Default

N.

LSB_ESUB_METHOD**Syntax**

LSB_ESUB_METHOD="*application_name* [*application_name*] ..."

Description

Specifies a mandatory **esub** or **esub** that applies to all job submissions. **LSB_ESUB_METHOD** lists the names of the application-specific **esub** or **esub** executables used in addition to any executables specified by the **bsub -a** option.

For example, **LSB_ESUB_METHOD="dce fluent"** runs `LSF_SERVERDIR/esub.dce` and `LSF_SERVERDIR/esub.fluent` for all jobs submitted to the cluster, then runs `LSF_SERVERDIR/esub.dce` and `LSF_SERVERDIR/esub.fluent` after the jobs are submitted. These **esubs** and **esubs** define, respectively, DCE as the mandatory security system and FLUENT as the mandatory application for all jobs. However, the purpose of an **esub** program might be different than its corresponding **esub** program because **esub** is run before the job is submitted, while **esub** is run after the job is submitted.

LSB_ESUB_METHOD can also be defined as an environment variable.

The value of **LSB_ESUB_METHOD** must correspond to an actual **esub** or **esub** file. For example, to use **LSB_ESUB_METHOD=fluent**, at least one of the `esub.fluent` file or the `esub.fluent` file must exist in `LSF_SERVERDIR`.

The name of the **esub** or **esub** program must be a valid file name. Valid file names contain only alphanumeric characters, underscore (`_`) and hyphen (`-`).

Restriction: The name `esub.user` is reserved. Do not use the name `esub.user` for an application-specific `esub`. This also applies to the **esub.user** file.

The master `esub` (**mesub**) uses the name you specify to invoke the appropriate **esub** or **esub** program. The **esub** and **esub.application_name** programs (or **esub** and **esub.application_name** programs) must be located in `LSF_SERVERDIR`.

LSF does not detect conflicts based on **esub** or **esub** names. For example, if **LSB_ESUB_METHOD="openmpi"** and `bsub -a pvm` is specified at job submission, the job could fail because these `esubs` define two different types of parallel job handling.

Default

Not defined. LSF does not apply a mandatory **esub** to jobs submitted to the cluster.

LSB_ESUB_SINGLE_QUOTE

Syntax

`LSB_ESUB_SINGLE_QUOTE=Y | y | N | n`

Description

If set to `Y` or `y`, the **LSB_SUB_PROJECT_NAME** values written to **\$LSB_SUB_PARM_FILE** for **esub** processes are enclosed in single quotation marks (`'`). This prevents the shell from processing certain meta characters in parameter values such as `$`.

If set to `N` or `n`, the **LSB_SUB_PROJECT_NAME** values written to **\$LSB_SUB_PARM_FILE** for **esub** processes are enclosed in double quotation marks (`"`).

Default

`N`.

LSB_ESWITCH_METHOD

Syntax

`LSB_ESWITCH_METHOD="application_name [application_name] ..."`

Description

Specifies a mandatory **eswitch** that applies to all job switch requests. **LSB_ESWITCH_METHOD** lists the names of the application-specific **eswitch** executable files used in addition to any executable files that are specified by the **bsub -a** option.

For example, **LSB_ESWITCH_METHOD="dce fluent"** runs `LSF_SERVERDIR/eswitch.dce` and `LSF_SERVERDIR/eswitch.fluent` for all job switch requests. These **eswitch** files define, respectively, DCE as the mandatory security system and FLUENT as the mandatory application for all job switch requests.

LSB_ESWITCH_METHOD can also be defined as an environment variable.

The value of **LSB_ESWITCH_METHOD** must correspond to an actual **eswitch** file. For example, to use **LSB_ESWITCH_METHOD=fluent**, the `eswitch.fluent` file must exist in the `LSF_SERVERDIR` directory.

LSF first invokes the executable file named **eswitch** (without *.application_name* in the file name) if it exists in the **LSF_SERVERDIR** directory. LSF then invokes the mandatory executable files specified in the **LSB_ESWITCH_METHOD** parameter. Finally, LSF invokes any application-specific **eswitch** executable files (with *.application_name* in the file name) specified by the **-a** option.

The name of the **eswitch** program must be a valid file name. Valid file names contain only alphanumeric characters, underscore (`_`) and hyphen (`-`).

Restriction: The name `eswitch.user` is reserved. Do not use the name `eswitch.user` for an application-specific **eswitch**.

LSF does not detect conflicts based on **eswitch** names. For example, if **LSB_ESWITCH_METHOD="openmpi"** and `bswitch -a pvm` is specified at job submission, the switch request could fail because these **eswitch** programs define two different types of parallel job handling.

Default

Not defined. LSF does not apply a mandatory **eswitch** to job switch requests.

LSB_EVENTS_FILE_KEEP_OPEN

Syntax

LSB_EVENTS_FILE_KEEP_OPEN=Y|N

Description

Windows only.

Specify Y to open the events file once, and keep it open always.

Specify N to open and close the events file each time a record is written.

Default

Y

LSB_EXCLUDE_HOST_PERIOD

Syntax

LSB_EXCLUDE_HOST_PERIOD=*mbd_sleep_time_units*

Description

Specifies the amount of time for hosts in a job's host exclusion list to remain excluded from that job (that is, the job does not run on these excluded hosts). This parameter indicates the number of **mbatchd** sleep time units (**MBD_SLEEP_TIME**) that a host remains excluded from a job. After this period of time, the hosts are no longer excluded and the job can run on the host again.

For example, if **MBD_SLEEP_TIME** is 10 seconds and **LSB_EXCLUDE_HOST_PERIOD** is 2, hosts that were excluded from a job remain excluded for 20 seconds.

If this parameter is not set, the hosts remain excluded indefinitely until **mbatchd** restarts.

This parameter does not apply to the MultiCluster job lease model.

Default

Not defined.

LSB_FANOUT_TIMEOUT_PER_LAYER

Syntax

LSB_FANOUT_TIMEOUT_PER_LAYER=time_seconds

Description

Controls how long **sbatchd** waits until the next **sbatchd** replies. Can also be set as an environment variable, which takes precedence.

Note: This feature is not currently supported on Windows.

Default

20 seconds.

LSB_FORK_JOB_REQUEST

Syntax

LSB_FORK_JOB_REQUEST=Y|N

Description

This parameter is enabled by default to improve **mbatchd** response time after **mbatchd** is started or restarted (including parallel restart) and has finished replaying events. A child **mbatchd** process is forked to sync cluster state information to **mbschd** after events have been replayed.

Default

Y

LSB_GPD_CLUSTER

Syntax

LSB_GPD_CLUSTER="*cluster_name* [*cluster_name* ...]"

Description

Defines the names of clusters whose master hosts start **gpolicyd** for global fairshare policy among multiple clusters. **LSB_GPD_CLUSTER** must be configured for every cluster participating in global fairshare.

Default

Not defined.

LSB_GPD_PORT

Syntax

LSB_GPD_PORT=port_number

Description

TCP service port for communication with **gpolicyd**. **LSB_GPD_PORT** must be configured for every cluster participating in global fairshare.

Default

Not defined.

LSB_GPU_AUTOBOOST (Obsolete)**Syntax**

LSB_GPU_AUTOBOOST=y|Y|n|N

Description

Obsolete. As of LSF 10.1 Fix Pack 4, LSF synchronizes the GPU auto-boost to resolve any issues that previously required disabling the auto-boost. Nvidia K80 provides the auto boost feature by default. This is designed for applications that use a single GPU. For multi-GPU jobs, this results in different GPUs running at different speeds. By default, LSF enables auto boost. LSF can switch off the auto boost for jobs with exclusive thread/process multi-GPU requirements.

If LSB_GPU_AUTOBOOST is set to N, LSF disables auto boost for jobs that require multiple exclusive mode GPUs whether these GPUs are located on one host or multiple hosts. If a job requires shared mode GPUs and exclusive mode GPUs at the same time, LSF does not turn off auto boost for this type of job. When this job is done or exited, LSF restores the original auto boost status.

You can override this parameter with the following job-level environment variable:

GPU_AUTOBOOST_FOR_JOB=y|Y|n|N

Note: LSF ignores the auto boost configuration change while **sbatchd** is running. Because Nvidia NVML library APIs have memory leak problems, **sbatchd** cannot call the Nvidia NVML library periodically to get the newest auto boost configuration. If an LSF administrator needs to change the GPU auto boost configuration on one host, clean all the jobs on the host, modify the GPU auto boost configuration, then restart **sbatchd**.

Default

y

LSB_GPU_NEW_SYNTAX

Enables the **bsub -gpu** option to submit jobs that require GPU resources. Also enables the specification of GPU resources with the **GPU_REQ** parameter in the `lsb.queues` and `lsb.applications` file or the default GPU requirement configured in the **LSB_GPU_REQ** parameter in the `lsf.conf` file.

Syntax

LSB_GPU_NEW_SYNTAX = Y | y | N | n | extend

Description

When this parameter is enabled (that is, set as Y or extend), you can use the **bsub -gpu** command, **GPU_REQ** parameter (in the `lsb.queues` and `lsb.applications` file), or **LSB_GPU_REQ** parameter (in the `lsf.conf` file) to specify your GPU requirements. Use these options and parameters to specify GPU requirements instead of using the **bsub -R** command or **RES_REQ** parameter in the `lsb.queues` file.

When the **LSB_GPU_NEW_SYNTAX=Y** parameter is set, you can specify the `j_exclusive`, `mode`, `mps`, and `num` GPU requirements with the **bsub -gpu** option. You can also specify GPU resource requirements with the **GPU_REQ** parameter in a queue (`lsb.queues` file), application profile (`lsb.applications` file), or a default GPU requirement with the **LSB_GPU_REQ** parameter (`lsf.conf` file). The resource requirements of your job submission cannot use the legacy GPU resources (`ngpus_shared`, `ngpus_excl_t`, `ngpus_excl_p`) as job resource requirements.

When the **LSB_GPU_NEW_SYNTAX=extend** parameter is set, you can specify the `gmem`, `gmodel`, `gtile`, and `nvlink` GPU requirements with the **bsub -gpu** option, in addition to the GPU requirements when

LSB_GPU_NEW_SYNTAX=Y is set (that is, `j_exclusive`, `mode`, `mps`, and `num` GPU requirements). You can also specify GPU resource requirements with the **GPU_REQ** parameter in a queue (`lsb.queues` file), application profile (`lsb.applications` file), or a default GPU requirement with the **LSB_GPU_REQ** parameter (`lsf.conf` file). The resource requirements of your job submission cannot use the legacy GPU resources (`ngpus_shared`, `ngpus_excl_t`, `ngpus_excl_p`) as job resource requirements.

When the **LSB_GPU_NEW_SYNTAX=N** parameter is set or the parameter is not specified, you must use the GPU resources `ngpus_shared`, `ngpus_excl_t` and `ngpus_excl_p` as GPU resource requirements.

Job-level GPU requirements override the application profile level, which overrides queue level, which overrides cluster level configuration.

There are some situations where you may want to use the **bsub -R** command option to specify GPU requirements instead of the **bsub -gpu** command option. These complex situations include the need to use compound or alternative resource requirements to specify your GPU requirements. For example, a GPU job might be available to run on either K40 or K80 GPUs. You cannot use **bsub -gpu** syntax to specify this type of selection and must instead use the **bsub -R** syntax to specify this selection using alternative resource requirements.

Since `j_exclusive`, `mode`, and `mps` GPU requirements are not supported in the **bsub -R** syntax (including the **RES_REQ** parameter), you must use the **bsub -gpu** syntax (including the **GPU_REQ** or **LSB_GPU_REQ** parameters). For these three GPU requirements, you can use the **bsub -R** syntax and the **bsub -gpu** syntax in the same job submission.

If you use the **bsub -R** syntax with the **bsub -gpu** syntax for any other GPU requirement at the job, application profile, or queue level (including `gmem`, `gmodel`, `gtile`, or `nvlink`), LSF rejects the job submission.

Setting the **LSB_GPU_NEW_SYNTAX** parameter to `Y` or `extend` also enables GPU resource preemption if you also configure the **PREEMPTABLE_RESOURCES** parameter in the `lsb.params` file to include the `ngpus_physical` resource. GPU jobs must be either using `exclusive_process` mode or have `j_exclusive=yes` set to be preempted by other GPU jobs. Non-GPU jobs cannot preempt GPU jobs.

Note: When LSF is running on RHEL systems, RHEL Version 7, or later, is required to support the **LSB_GPU_NEW_SYNTAX** parameter.

Default

Not defined. The job submission option `bsub -gpu` and the GPU requirements in the **GPU_REQ** parameter are rejected.

See also

- **GPU_REQ**
- **LSB_GPU_REQ**
- **bsub -gpu**

LSB_GPU_REQ

Specify GPU requirements together in one statement.

Syntax

```
LSB_GPU_REQ = "[num=num_gpus[/task | host]] [:mode=shared | exclusive_process]
[:mps=yes[, shared] | no | per_socket[, shared] | per_gpu[, shared]] [:j_exclusive=yes | no]
[:aff=yes | no] "
```

Description

The **LSB_GPU_NEW_SYNTAX=extend** or **LSB_GPU_NEW_SYNTAX=Y** parameter must be set in the `lsf.conf` file to specify the `j_exclusive`, `mode`, `mps`, and `num` GPU requirements in the **LSB_GPU_REQ** parameter.

When the **LSB_GPU_NEW_SYNTAX=extend** parameter is set, the resource requirements of your job cannot use the legacy GPU resources (`ngpus_shared`, `ngpus_excl_t`, `ngpus_excl_p`) as job resource requirements.

The **LSB_GPU_REQ** parameter takes the following arguments:

num=num_gpus[/task | host]

The number of physical GPUs required by the job. By default, the number is per host. You can also specify that the number is per task by specifying `/task` after the number.

If you specified that the number is per task, the configuration of the `ngpus_physical` resource in the `lsb.resources` file is set to `PER_TASK`, or the **RESOURCE_RESERVE_PER_TASK=Y** parameter is set in the `lsb.params` file, this number is the requested count per task.

mode=shared | exclusive_process

The GPU mode when the job is running, either `shared` or `exclusive_process`. The default mode is `shared`.

The `shared` mode corresponds to the NVIDIA DEFAULT compute mode. The `exclusive_process` mode corresponds to the NVIDIA EXCLUSIVE_PROCESS compute mode.

mps=yes[,shared] | per_socket[,shared] | per_gpu[,shared] | no

Enables or disables the NVIDIA Multi-Process Service (MPS) for the GPUs that are allocated to the job. Using MPS effectively causes the EXCLUSIVE_PROCESS mode to behave like the DEFAULT mode for all MPS clients. MPS always allows multiple clients to use the GPU through the MPS server.

MPS is useful for both shared and exclusive process GPUs, and allows more efficient sharing of GPU resources and better GPU utilization. See the NVIDIA documentation for more information and limitations.

When using MPS, use the EXCLUSIVE_PROCESS mode to ensure that only a single MPS server is using the GPU, which provides additional insurance that the MPS server is the single point of arbitration between all CUDA process for that GPU.

You can also enable MPS daemon sharing by adding the **share** keyword with a comma and no space (for example, `mps=yes,shared` enables MPS daemon sharing on the host). If sharing is enabled, all jobs that are submitted by the same user with the same resource requirements share the same MPS daemon on the host, socket, or GPU.

LSF starts MPS daemons on a per-host, per-socket, or per-GPU basis depending on value of the **mps** keyword:

- If `mps=yes` is set, LSF starts one MPS daemon per host per job.

When `share` is enabled (that is, if `mps=yes,shared` is set), LSF starts one MPS daemon per host for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon on the host.
- If `mps=per_socket` is set, LSF starts one MPS daemon per socket per job. When enabled with `share` (that is, if `mps=per_socket,shared` is set), LSF starts one MPS daemon per socket for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon for the socket.
- If `mps=per_gpu` is set, LSF starts one MPS daemon per GPU per job. When enabled with `share` (that is, if `mps=per_gpu,shared` is set), LSF starts one MPS daemon per GPU for all jobs that are submitted by the same user with the same resource requirements. These jobs all use the same MPS daemon for the GPU.

Note: Using EXCLUSIVE_THREAD mode with MPS is not supported and might cause unexpected behavior.

j_exclusive=yes | no

Specifies whether the allocated GPUs can be used by other jobs. When the mode is set to `exclusive_process`, the `j_exclusive=yes` option is set automatically.

aff=yes | no

Specifies whether to enforce strict GPU-CPU affinity binding. If set to no, LSF relaxes GPU affinity while maintaining CPU affinity. By default, `aff=yes` is set to maintain strict GPU-CPU affinity binding.

The `LSB_GPU_NEW_SYNTAX=extend` or `LSB_GPU_NEW_SYNTAX=Y` parameter must be defined in the `lsf.conf` file to enable the `-gpu` option and `GPU_REQ` parameter syntax. When the `LSB_GPU_NEW_SYNTAX=extend` parameter is set, you cannot use the GPU resources `ngpus_shared`, `ngpus_excl_t` and `ngpus_excl_p` as job resource requirements.

If the `GPU_REQ_MERGE` parameter is defined as Y or y in the `lsb.params` file and a GPU requirement is specified at multiple levels (at least two of the default cluster, queue, application profile, or job level requirements), each option of the GPU requirement is merged separately. Job level overrides application level, which overrides queue level, which overrides the default cluster GPU requirement. For example, if the mode option of the GPU requirement is defined on the `-gpu` option, and the mps option is defined in the queue, the mode of job level and the mps value of queue is used.

If the `GPU_REQ_MERGE` parameter is not defined as Y or y in the `lsb.params` file and a GPU requirement is specified at multiple levels (at least two of the default cluster, queue, application profile, or job level requirements), the entire GPU requirement string is replaced. The entire job level GPU requirement string overrides application level, which overrides queue level, which overrides the default GPU requirement.

The `esub` parameter `LSB_SUB4_GPU_REQ` modifies the value of the `-gpu` option.

LSF selects the GPU that meets the topology requirement first. If the GPU mode of the selected GPU is not the requested mode, LSF changes the GPU to the requested mode. For example, if LSF allocates an `exclusive_process` GPU to a job that needs a shared GPU, LSF changes the GPU mode to `shared` before the job starts and then changes the mode back to `exclusive_process` when the job finishes.

The GPU requirements are converted to `rusage` resource requirements for the job. For example, `num=2` is converted to `rusage[ngpus_physical=2]`. Use the `bjobs`, `bhist`, and `bacct` commands to see the merged resource requirement.

There might be complex GPU requirements that the `bsub -gpu` option and `GPU_REQ` parameter syntax cannot cover, including compound GPU requirements (for different GPU requirements for jobs on different hosts, or for different parts of a parallel job) and alternate GPU requirements (if more than one set of GPU requirements might be acceptable for a job to run). For complex GPU requirements, use the `bsub -R` command option, or the `RES_REQ` parameter in the `lsb.applications` or `lsb.queues` file to define the resource requirement string.

Note: You can define the mode, `j_exclusive`, and mps options only with the `-gpu` option, the `LSB_GPU_REQ` parameter in the `lsf.conf` file, or the `GPU_REQ` parameter in the `lsb.queues` or `lsb.applications` files. You cannot use these options with the `rusage` resource requirement string in the `bsub -R` command option or the `RES_REQ` parameter in the `lsb.queues` or `lsb.applications` files.

Default

Not defined

See also

- `GPU_REQ`
- `LSB_GPU_NEW_SYNTAX`
- `bsub -gpu`

LSB_GPU_POWEROFF_DURATION**Syntax**

`LSB_GPU_POWEROFF_DURATION=seconds`

Description

A GPU consumes significant power even when it idles. LSF can decrease the GPU power consumed if GPU is not in use within a specified time. LSB_GPU_POWEROFF_DURATION enables this feature to specify the minimum time before LSF can power off a GPU if the GPU is idle.

LSF tries to allocate the GPU that is not running as "MIN power limit" mode. If the number of GPUs that are in "MAX power limit" mode are not enough, LSF allocates the GPUs that are in "MIN power limit" mode and switches those GPUs to run in "MAX power limit" mode.

LSF allocates the GPU flow as follows:

- GPUs are in the same PCI.
- Check if the MAX power limit mode GPUs meets job requirements. If they do, LSF does not allocate the MIN power limit mode GPUs first. If they do not meet the requirements, LSF allocates all the GPUs to the job, including both MAX power limit and MIN power limit mode GPUs.

If **sbatchd** is restarted, the GPU idle time is recalculated.

Specify a non-negative value as the minimum number of seconds that a GPU is idle before LSF powers it off.

If set to 0, LSF powers off GPUs immediately after the job finishes.

Nvidia K80 hardware supports auto boost and switch power limits. The NVML library must be Version 6.340 or newer.

Default

Not defined. LSF never powers off a GPU even when it is idle.

LSB_GSLA_DISPLAY_ALLOC_HOSTS**Syntax**

LSB_GSLA_DISPLAY_ALLOC_HOSTS=Y | y | N | n

Description

If set to Y or y, the **bsla** command displays information on guarantee hosts that are being used (allocated) from each guarantee pool for the guarantee SLA.

Default

N. The **bsla** command does not display the guarantee hosts that are being used for the guarantee SLA.

LSB_GSLA_PREFER_ADRSV_HOST**Syntax**

LSB_GSLA_PREFER_ADRSV_HOST=Y | y | N | n

Description

If set to Y or y, the guarantee SLA will first try to reserve hosts without advanced reservation. This ensures that advance reservation does not interfere with guarantee SLA job scheduling.

Default

N.

LSB_HJOB_PER_SESSION

Syntax

LSB_HJOB_PER_SESSION=*max_num*

Description

Specifies the maximum number of jobs that can be dispatched in each scheduling cycle to each host

Valid values

Any positive integer

Default

Not defined

Notes

LSB_HJOB_PER_SESSION is activated only if the **JOB_ACCEPT_INTERVAL** parameter is set to 0.

See also

JOB_ACCEPT_INTERVAL parameter in `lsb.params`

LSB_HMS_TIME_FORMAT

Syntax

LSB_HMS_TIME_FORMAT=*y|Y|n|N*

Description

When this parameter is set to Y or y, **bjobs** displays any times (but not time stamps) in the customized output in *hh:mm:ss* format.

The `cpu_used` field normally uses one decimal place in the **bjobs** output, but if you specify **LSB_HMS_TIME_FORMAT=Y**, **bjobs** rounds up this field to the next second. For example, if the `cpu_used` field is 0.2 seconds, specifying **LSB_HMS_TIME_FORMAT=Y** rounds this number up to 00:00:01.

For the `runtime_limit` field, if the **ABS_RUNLIMIT** parameter is defined as Y in the `lsb.params` file and you specify **LSB_HMS_TIME_FORMAT=Y**, **bjobs** does not display the host name. That is, if **ABS_RUNLIMIT=Y** is defined in the `lsb.params` file, **bjobs** normally displays the host name after the runtime limit (for example, 3.0/hostA). If you specify **LSB_HMS_TIME_FORMAT=Y**, **bjobs** displays the runtime limit without the host name (for example, 00:03:00). This also applies if the **ABS_RUNLIMIT** parameter is defined as Y in the `lsb.applications` file and you specify **LSB_HMS_TIME_FORMAT=Y** to a job that is submitted to an application profile with **ABS_RUNLIMIT=Y**.

Specifying the **bjobs -hms** option overrides the **LSB_HMS_TIME_FORMAT** environment variable, which overrides the **LSB_HMS_TIME_FORMAT** parameter setting in the `lsf.conf` file.

This option applies only to output for the **bjobs -o** and **bjobs -o -json** commands for customized output. This option has no effect when run with **bjobs** without the `-o` option.

Default

N.

LSB_HUGETLB

Syntax

LSB_HUGETLB=Y|N

Description

The information regarding which virtual memory page maps to which physical memory page is kept in a data structure named Page Table. Most architectures use a fast lookup cache named Translation Lookaside Buffer (TLB). Consequently, TLB misses bring additional performance costs, so it is important to reduce TLB misses.

In advanced architectures like x86 or IA64, huge page size is supported (e.g., 2 Mb and 4 Mb sizes). The number of memory pages can be reduced through implementing the huge page size, which leads to decreased TLB misses and improved performance for processes like forked child, etc.

To configure huge page memory:

1. Check the support and configuration of huge page size:

```
cat /proc/meminfo | grep Huge
```

The output of `cat /proc/meminfo` will include lines such as:

```
HugePages_Total: vvv
```

```
HugePages_Free: www
```

```
HugePages_Rsvd: xxx
```

```
HugePages_Surp: yyy
```

```
Hugepagesize: zzz kB
```

Where:

- **HugePages_Total** is the size of the pool of huge pages.
- **HugePages_Free** is the number of huge pages in the pool that are not yet allocated.
- **HugePages_Rsvd** is short for "reserved" and is the number of huge pages for which a commitment to allocate from the pool has been made, though no allocation has yet been made. Reserved huge pages guarantee that an application can allocate a huge page from the pool of huge pages at fault time.
- **HugePages_Surp** is short for "surplus" and is the number of huge pages in the pool above the value in `/proc/sys/vm/nr_hugepages`. The maximum number of surplus huge pages is controlled by `/proc/sys/vm/nr_overcommit_hugepages`.

2. Configure the number of huge size pages:

- To set the number of huge pages using `/proc` entry:

```
# echo 5 > /proc/sys/vm/nr_hugepages
```

- To set the number of huge pages using `sysctl`:

```
# sysctl -w vm.nr_hugepages=5
```

- To make the change permanent, add the following line to the file `/etc/sysctl.conf`:

```
# echo "vm.nr_hugepages=5" >> /etc/sysctl.conf
```

This file is used during the boot process.

You must reboot to allocate the number of hugepages needed. This is because hugepages requires large areas of contiguous physical memory. Over time, physical memory may be mapped and allocated to pages. Therefore, the physical memory can become fragmented.

Default

N

LSB_IGNORE_ELIM_RES**Syntax**

LSB_IGNORE_ELIM_RES=Y | y | N | n

Description

Specifies whether LSF ignores dynamic shared resources reported by the **elim** commands when checking the reservation status of a preempting job.

Used when validating reservations for preempting jobs. During the preemption phase, dynamic shared resources that are slow to update can incorrectly invalidate reservations.

Setting LSB_IGNORE_ELIM_RES=Y enables LSF to ignore the dynamic shared resources reported by the **elim** commands when checking the reservation status of a preempting job. If higher priority preempting jobs have taken the reserved resources, the job will not preempt again until the **elim** has updated the resources.

Default

N

LSB_INDEX_BY_JOB**Syntax**

LSB_INDEX_BY_JOB="JOBNAME"

Description

When set to JOBNAME, creates a job index of job names. Define when using job dependency conditions (**bsub -w**) with job names to optimize job name searches.

Valid values

JOBNAME

Default

Not defined (job index is not created).

LSB_INTERACT_MSG_ENH**Syntax**

LSB_INTERACT_MSG_ENH=y | Y

Description

If set, enables enhanced messaging for interactive batch jobs. To disable interactive batch job messages, set LSB_INTERACT_MSG_ENH to any value other than y or Y; for example, LSB_INTERACT_MSG_ENH=N.

Default

Not defined

See also**LSB_INTERACT_MSG_INTVAL****LSB_INTERACT_MSG_INTVAL****Syntax**LSB_INTERACT_MSG_INTVAL=*time_seconds***Description**

Specifies the update interval in seconds for interactive batch job messages.

LSB_INTERACT_MSG_INTVAL is ignored if LSB_INTERACT_MSG_ENH is not set.

Job information that LSF uses to get the pending or suspension reason is updated according to the value of PEND_REASON_UPDATE_INTERVAL in `lsb.params`.

Default

Not defined. If LSB_INTERACT_MSG_INTVAL is set to an incorrect value, the default update interval is 60 seconds.

See also**LSB_INTERACT_MSG_ENH****LSB_JOB_CPULIMIT**

Determines whether the CPU limit is a per-process limit enforced by the OS or whether it is a per-job limit enforced by LSF.

SyntaxLSB_JOB_CPULIMIT=*y* | *n***Description**

- The per-process limit is enforced by the OS when the CPU time of one process of the job exceeds the CPU limit.
- The per-job limit is enforced by LSF when the total CPU time of all processes of the job exceed the CPU limit.

This parameter applies to CPU limits set when a job is submitted with **bsub -c**, and to CPU limits set for queues by the **CPULIMIT** parameter in the `lsb.queues` file.

LSF-enforced per-job limit

When the sum of the CPU time of all processes of a job exceed the CPU limit, LSF sends a SIGXCPU signal (if this signal is supported by the operating system) from the operating system to all processes belonging to the job, then SIGINT, SIGTERM and SIGKILL. The interval between signals is 10 seconds by default. The time interval between SIGXCPU, SIGINT, SIGKILL, SIGTERM can be configured with the parameter **JOB_TERMINATE_INTERVAL** in the `lsb.params` file.

Restriction:

SIGXCPU is not supported by Windows.

OS-enforced per process limit

When one process in the job exceeds the CPU limit, the limit is enforced by the operating system. For more details, refer to your operating system documentation for `setrlimit()`.

The setting of the **LSB_JOB_CPULIMIT** parameter has the following effect on how the limit is enforced:

LSB_JOB_CPULIMIT	LSF per-job limit	OS per-process limit
y	Enabled	Disabled
n	Disabled	Enabled
Not defined	Enabled	Enabled

Default

Not defined

Notes

To make changes to the **LSB_JOB_CPULIMIT** parameter take effect, use the command **badmin hrestart** all to restart all sbatchds in the cluster.

Changing the default Terminate job control action

You can define a different terminate action in the `lsb.queues` file with the parameter **JOB_CONTROLS** if you do not want the job to be killed. For more details on job controls, see *Administering IBM Spectrum LSF*.

Limitations

If a job is running and the parameter is changed, LSF is not able to reset the type of limit enforcement for running jobs.

- If the parameter is changed from per-process limit enforced by the OS to per-job limit enforced by LSF (**LSB_JOB_CPULIMIT=n** changed to **LSB_JOB_CPULIMIT=y**), both per-process limit and per-job limit affect the running job. This means that signals may be sent to the job either when an individual process exceeds the CPU limit or the sum of the CPU time of all processes of the job exceed the limit. A job that is running may be killed by the OS or by LSF.
- If the parameter is changed from per-job limit enforced by LSF to per-process limit enforced by the OS (**LSB_JOB_CPULIMIT=y** changed to **LSB_JOB_CPULIMIT=n**), the job is allowed to run without limits because the per-process limit was previously disabled.

See also

`lsb.queues`, **bsub**, **JOB_TERMINATE_INTERVAL** in `lsb.params`, **LSB_MOD_ALL_JOBS**

LSB_JOBID_DISP_LENGTH

Syntax

`LSB_JOBID_DISP_LENGTH=integer`

Description

By default, LSF commands **bjobs** and **bhist** display job IDs with a maximum length of 7 characters. Job IDs greater than 9999999 are truncated on the left.

When `LSB_JOBID_DISP_LENGTH=10`, the width of the JOBID column in **bjobs** and **bhist** increases to 10 characters.

If **LSB_BJOBS_FORMAT** is defined (in `lsf.conf` or as a runtime environment variable) or **bjobs -o** is run to include the JOBID column, and either of these specify a column width for the JOBID column, those specifications override the **LSB_JOBID_DISP_LENGTH** value. If there is no column width specified for the JOBID column, the **LSB_JOBID_DISP_LENGTH** value applies.

Valid values

Specify an integer between 7 and 10.

Default

Not defined. LSF uses the default 7-character length for job ID display.

LSB_JOB_MEMLIMIT

Determines whether the memory limit is a per-process limit enforced by the OS or whether it is a per-job limit enforced by LSF.

Syntax

LSB_JOB_MEMLIMIT=y | n

Description

- The per-process limit is enforced by the OS when the memory allocated to one process of the job exceeds the memory limit.
- The per-job limit is enforced by LSF when the sum of the memory allocated to all processes of the job exceeds the memory limit.

This parameter applies to memory limits set when a job is submitted with **bsub -M mem_limit**, and to memory limits set for queues with the **MEMLIMIT** parameter in the `lsb.queues` file.

The setting of the **LSB_JOB_MEMLIMIT** parameter has the following effect on how the limit is enforced:

LSB_JOB_MEMLIMIT	LSF-enforced per-job limit	OS-enforced per-process limit
y	Enabled	Disabled
n or not defined	Disabled	Enabled

When the **LSB_JOB_MEMLIMIT** parameter is Y, the LSF-enforced per-job limit is enabled, and the OS-enforced per-process limit is disabled.

When **LSB_JOB_MEMLIMIT** is N or not defined, the LSF-enforced per-job limit is disabled, and the OS-enforced per-process limit is enabled.

LSF-enforced per-job limit

When the total memory allocated to all processes in the job exceeds the memory limit, LSF sends the following signals to kill the job: SIGINT, SIGTERM, then SIGKILL. The interval between signals is 10 seconds by default.

On UNIX, the time interval between SIGINT, SIGKILL, SIGTERM can be configured with the parameter `JOB_TERMINATE_INTERVAL` in `lsb.params`.

OS-enforced per process limit

When the memory allocated to one process of the job exceeds the memory limit, the operating system enforces the limit. LSF passes the memory limit to the operating system. Some operating systems apply the memory limit to each process, and some do not enforce the memory limit at all.

OS memory limit enforcement is only available on systems that support `RLIMIT_RSS` for `setrlimit()`.

The following operating systems do not support the memory limit at the OS level and the job is allowed to run without a memory limit:

- Windows
- Sun Solaris 2.x

Default

Not defined. Per-process memory limit enforced by the OS; per-job memory limit enforced by LSF disabled

Notes

To make changes to the **LSB_JOB_MEMLIMIT** parameter take effect, use the command **badmin hrestart all** to restart all sbatchds in the cluster.

If **LSB_JOB_MEMLIMIT** is set, it overrides the setting of the parameter **LSB_MEMLIMIT_ENFORCE**. The parameter **LSB_MEMLIMIT_ENFORCE** is ignored.

The difference between **LSB_JOB_MEMLIMIT** set to y and **LSB_MEMLIMIT_ENFORCE** set to y is that with **LSB_JOB_MEMLIMIT**, only the per-job memory limit enforced by LSF is enabled. The per-process memory limit enforced by the OS is disabled. With **LSB_MEMLIMIT_ENFORCE** set to y, both the per-job memory limit enforced by LSF and the per-process memory limit enforced by the OS are enabled.

Changing the default Terminate job control action

You can define a different Terminate action in the `lsb.queues` file with the parameter **JOB_CONTROLS** if you do not want the job to be killed. For more details on job controls, see *Administering IBM Spectrum LSF*.

Limitations

If a job is running and the parameter is changed, LSF is not able to reset the type of limit enforcement for running jobs.

- If the parameter is changed from per-process limit enforced by the OS to per-job limit enforced by LSF (**LSB_JOB_MEMLIMIT=n** or not defined changed to **LSB_JOB_MEMLIMIT=y**), both per-process limit and per-job limit affect the running job. This means that signals may be sent to the job either when the memory allocated to an individual process exceeds the memory limit or the sum of memory allocated to all processes of the job exceed the limit. A job that is running may be killed by LSF.
- If the parameter is changed from per-job limit enforced by LSF to per-process limit enforced by the OS (**LSB_JOB_MEMLIMIT=y** changed to **LSB_JOB_MEMLIMIT=n** or not defined), the job is allowed to run without limits because the per-process limit was previously disabled.

See also

LSB_MEMLIMIT_ENFORCE, **LSB_MOD_ALL_JOBS**, `lsb.queues`, **bsub**, **JOB_TERMINATE_INTERVAL** in `lsb.params`

LSB_JOB_OUTPUT_LOGGING**Syntax**

`LSB_JOB_OUTPUT_LOGGING=Y | N`

Description

Determines whether jobs write job notification messages to the logfile.

Default

Not defined (jobs do not write job notification messages to the logfile).

LSB_JOB_REPORT_MAIL

Syntax

LSB_JOB_REPORT_MAIL=Y | N | ERROR

Description

If you do not want **sbatchd** to send mail when the job is done, set this parameter to N before submitting the job. This parameter only affects email sent by **sbatchd**.

If you want **sbatchd** to send mail only when the job exits (that is, when the job is under Exit status), set this parameter to ERROR before submitting the job. This parameter only affects email sent by **sbatchd**. This ensures that an email notification is only sent on a job error.

When the administrator sets **LSB_JOB_REPORT_MAIL** in `lsf.conf`, email notification for all jobs is disabled. All **sbatchds** must be restarted on all hosts. However, end users can set the value for **LSB_JOB_REPORT_MAIL** in the job submission environment to disable email notification for only that particular job and not email for all jobs. In this case, there is no need to restart **sbatchd**.

Default

Not defined.

LSB_JOB_SCRIPT_TRAP_SIGNALS

Syntax

LSB_JOB_SCRIPT_TRAP_SIGNALS=*signal_name* ...

Description

Specifies a list of the names of signals that are trapped by the job scripts.

This parameter prevents the specified signals from killing the job script process.

By default, the job scripts trap the SIGTERM, SIGUSR1, SIGUSR2, SIGINT, and SIGHUP signals, so you do not have to define these signals in this parameter.

Because the job scripts cannot trap the SIGSTOP and SIGKILL signals, these values are not valid.

Valid values

A space-separated list of signal names. The first 31 signals are valid (from SIGHUP to SIGSYS), except for SIGSTOP and SIGKILL.

Default

Undefined. The job script does not trap any additional signals except SIGTERM, SIGUSR1, SIGUSR2, SIGINT, and SIGHUP.

LSB_JOB_TMPDIR

Syntax

LSB_JOB_TMPDIR=*directory*

Description

Specifies the path and directory for temporary job output.

When `LSB_JOB_TMPDIR` is defined in `lsf.conf`, LSF uses the directory specified by `LSB_JOB_TMPDIR` on the execution host when a job is started and creates a job-specific temporary directory as a subdirectory of the `LSB_JOB_TMPDIR` directory.

The name of the job-specific temporary directory has the following format:

- For regular jobs:
 - Unix: `$LSB_JOB_TMPDIR/jobID.tmpdir`
 - Windows: `%LSB_JOB_TMPDIR%\jobID.tmpdir`
- For array jobs:
 - Unix: `$LSB_JOB_TMPDIR/arrayID_arrayIndex.tmpdir`
 - Windows: `%LSB_JOB_TMPDIR%\arrayID_arrayIndex.tmpdir`

On UNIX, the directory has the permission `0700` and is owned by the execution user.

After adding `LSB_JOB_TMPDIR` to `lsf.conf`, use **admin hrestart all** to reconfigure your cluster.

If `LSB_SET_TMPDIR=Y`, the environment variable `TMPDIR` will be set to the job-specific temporary directory.

Valid values

Specify any valid path up to a maximum length of 256 characters. The 256 character maximum path length includes the temporary directories and files that the system creates as jobs run. The path that you specify for `LSB_JOB_TMPDIR` should be as short as possible to avoid exceeding this limit.

The path can include the `%H` dynamic pattern, which is automatically replaced with the first execution host of the job.

UNIX

Specify an absolute path. For example:

```
LSB_JOB_TMPDIR=/usr/share/lsf_tmp
```

Windows

Specify a UNC path or a path with a drive letter. For example:

```
LSB_JOB_TMPDIR=\\HostA\temp\lsf_tmp
```

```
LSB_JOB_TMPDIR=D:\temp\lsf_tmp
```

Temporary directory for tasks launched by blaunch

By default, LSF creates the job-specific temporary directory only on the first execution host.

To create a job-specific temporary directory on each execution host, set **LSB_SET_TMPDIR=Y** so that the path of the job-specific temporary directory is available through the `TMPDIR` environment variable, or set **LSB_SET_TMPDIR** to a user-defined environment variable so that the path of the job-specific temporary directory is available through the user-defined environment variable.

Tasks launched through the **blaunch** distributed application framework make use of the job-specific temporary directory:

- When the job-specific temporary directory environment variable is set on the first execution host, the **blaunch** framework propagates this environment variable to all execution hosts when launching remote tasks
- The job RES or the task RES creates the job-specific temporary directory if it does not already exist before starting the job

- The directory created by the job RES or task RES has permission 0700 and is owned by the execution user
- If the job-specific temporary directory was created by the task RES, LSF deletes the directory and its contents when the task is complete
- If the job-specific temporary directory was created by the job RES, LSF deletes the directory and its contents when the job is done
- If the job-specific temporary directory is on a shared file system, it is assumed to be shared by all the hosts allocated to the **blaunch** job, so LSF *does not* remove the directories created by the job RES or task RES

Default

By default, LSB_JOB_TMPDIR is not enabled. If LSB_JOB_TMPDIR is not specified in `lsf.conf`, this parameter is defined as follows:

- On UNIX: \$TMPDIR or /tmp
- On Windows: %TMP%, %TEMP, or %SystemRoot%

LSB_JOBID_DISP_LENGTH**Syntax**

LSB_JOBID_DISP_LENGTH=*integer*

Description

By default, LSF commands `bjobs` and `bhist` display job IDs with a maximum length of 7 characters. Job IDs greater than 9999999 are truncated on the left.

When LSB_JOBID_DISP_LENGTH=10, the width of the JOBID column in **bjobs** and **bhist** increases to 10 characters.

If **LSB_BJOBS_FORMAT** is defined (in `lsf.conf` or as a runtime environment variable) or **bjobs -o** is run to include the JOBID column, and either of these specify a column width for the JOBID column, those specifications override the **LSB_JOBID_DISP_LENGTH** value. If there is no column width specified for the JOBID column, the **LSB_JOBID_DISP_LENGTH** value applies.

Valid values

Specify an integer between 7 and 10.

Default

Not defined. LSF uses the default 7-character length for job ID display.

LSB_JOBINFO_DIR**Syntax**

LSB_JOBINFO_DIR=*directory*

Description

Use this parameter to specify a directory for job information instead of using the default directory. If this parameter is specified, LSF directly accesses this directory to get the job information files.

By default, the job information directory is located in the LSF shared directory, which is in the same file system as the one used for logging events. In large scale clusters with millions of single jobs, there are several job files in the job information directory. The job information directory requires random read/write

operations on multiple job files simultaneously, while the event log directory has events appended to a single events file.

The LSB_JOBINFO_DIR directory must be the following:

- Owned by the primary LSF administrator
- Accessible from all hosts that can potentially become the master host
- Accessible from the master host with read and write permission
- Set for 700 permission

If the directory cannot be created, **mbatchd** will exit.

Note: Using the **LSB_JOBINFO_DIR** parameter requires draining the whole cluster.

Note: **LSB_JOBINFO_DIR** should be used for XL clusters. If it is configured for a non-XL cluster, all of the old job info directories must be copied to the new specified location.

Note: If **LSB_LOCALDIR** is configured, then **LSB_JOBINFO_DIR** will be ignored

Default

If this parameter is not set, it uses the following path from `lsb.params`:

```
$LSB_SHAREDIR/cluster_name/logdir/info
```

LSB_JSM_DEFAULT

Specifies the default value for the **bsub -j sm** option for CSM jobs.

Syntax

```
LSB_JSM_DEFAULT=y | n
```

Description

Run the **bsub -j sm** option at job submission time to override this default value.

Default

Not defined.

LSB_KEEP_SYSDEF_RLIMIT

Syntax

```
LSB_KEEP_SYSDEF_RLIMIT=y | n
```

Description

If resource limits are configured for a user in the SGI IRIX User Limits Database (ULDB) domain specified in `LSF_ULDB_DOMAIN`, and there is no domain default, the system default is honored.

If `LSB_KEEP_SYSDEF_RLIMIT=n`, and no resource limits are configured in the domain for the user and there is no domain default, LSF overrides the system default and sets system limits to unlimited.

Default

Not defined. No resource limits are configured in the domain for the user and there is no domain default.

LSB_KRB_AIX_LOAD_IBM_NAS

Syntax

LSB_KRB_AIX_LOAD_IBM_NAS=Y | N

Description

Use this parameter to enable the user Ticket Granting Ticket (TGT) forwarding feature without requiring open source client libraries, which is achieved by providing versions of the relevant LSF binary files (`bsub`, `krbrenwd`, and `sbatchd`) that dynamically link to the shared library provided in IBM's "Network Authentication Service 1.6.0.2 for AIX" package.

When set to Y on AIX platforms, LSF loads the files from `LSB_KRB_LIB_PATH`, otherwise LSF loads the required open source client libraries.

Default

N. LSF loads the required open source libraries.

See also

LSB_KRB_TGT_FWD

LSB_KRB_CHECK_INTERVAL

Syntax

LSB_KRB_CHECK_INTERVAL=*minutes*

Description

Set a time interval for how long `krbrenwd` and root `sbatchd` should wait before the next check. If this parameter is changed, restart `mbatchd/sbatchd`

Default

15 minutes

LSB_KRB_LIB_PATH

Syntax

LSB_KRB_LIB_PATH=path to krb5 lib

Description

Specify the library path that contains the krb5 libs:

- `libkrb5.so`
- `libcom_err.so`
- `libk5crypto.so`
- `libkrb5support.so`

You can configure multiple paths in this parameter. These paths can be blank, comma or semicolon separated. LSF will load the libs from these paths in the order you specified. Once loaded successfully, the search stops.

Default

On 32 bit platforms: `/lib, /usr/lib, /usr/local/lib`

On 64 bit platforms: `/lib64, /usr/lib64, /usr/local/lib64`

LSB_KRB_RENEW_MARGIN**Syntax**

`LSB_KRB_RENEW_MARGIN=minutes`

Description

Specify how long `krbrenewd` and `root sbatchd` have to renew a Ticket Granting Ticket (TGT) before it expires. If this parameter is changed, restart `mbatchd/sbatchd` to have it take effect.

Default

60 minutes

LSB_KRB_TGT_FWD**Syntax**

`LSB_KRB_TGT_FWD=Y|N`

Description

Use this parameter to control the user Ticket Granting Ticket (TGT) forwarding feature. When set to Y, user TGT is forwarded from the submission host to the execution host. `mbatchd` and/or `root sbatchd` do the required renewing along the way.

Default

N. (Do not forward user TGT during job submission.)

LSB_KRB_TGT_DIR**Syntax**

`LSB_KRB_TGT_DIR=directory`

Description

Specify a directory in which Ticket Granting Ticket (TGT) for a running job is stored. Each job or task will have its environment variable `KRB5CCNAME` pointing to the TGT file. Please note that this parameter controls the TGT location for running jobs, not for pending jobs on the `mbatchd` side. For pending jobs, TGTs are stored in the `mbatchd` info directory, along with their job log file.

LSF tries to find the directory to place user TGTs in the following order:

1. **LSB_KRB_TGT_DIR**
2. `/tmp` on execution host

Once LSF finds a valid directory, the search stops.

If **LSB_KRB_TGT_DIR** is not defined to `/tmp`, LSF will create a symbolic link in `/tmp` to point to the actual TGT file. The symlink name pattern is `lsf_krb5cc_${jid}_${some_suffix}`.

Default

Not defined

LSB_LOCALDIR

Syntax

LSB_LOCALDIR=*path*

Description

Enables duplicate logging.

Specify the path to a local directory that exists only on the first LSF master host. LSF puts the primary copies of the event and accounting log files in this directory. LSF puts the duplicates in **LSB_SHAREDIR**.

Important:

Always restart both **mbatchd** and **sbatchd** when modifying **LSB_LOCALDIR**.

Example

```
LSB_LOCALDIR=/usr/share/lsbatch/loginfo
```

Default

Not defined

See also

LSB_SHAREDIR, **EVENT_UPDATE_INTERVAL** in `lsb.params`

LSB_LOG_MASK_MBD

Syntax

LSB_LOG_MASK_MBD=*message_log_level*

Description

Specifies the logging level of error messages for LSF **mbatchd** only. This value overrides **LSB_LOG_MASK** for **mbatchd** only.

For example:

```
LSB_LOG_MASK_MBD=LOG_DEBUG
```

The valid log levels for this parameter are:

- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

Run **badmin mbdrestart** to make changes take effect.

Default

Not defined (logging level is controlled by LSB_LOG_MASK).

LSB_LOG_MASK_SBD**Syntax**

LSB_LOG_MASK_SBD=*message_log_level*

Description

Specifies the logging level of error messages for LSF **sbatchd** only. This value overrides LSF_LOG_MASK for **sbatchd** only.

For example:

```
LSB_LOG_MASK_SBD=LOG_DEBUG
```

The valid log levels for this parameter are:

- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

Run **badadmin hrestart** to make changes take effect.

Default

Not defined (logging level is controlled by LSF_LOG_MASK).

LSB_LOG_MASK_SCH**Syntax**

LSB_LOG_MASK_SCH=*message_log_level*

Description

Specifies the logging level of error messages for LSF **mbschd** only. This value overrides LSB_LOG_MASK for **mbschd** only.

For example:

```
LSB_LOG_MASK_SCHD=LOG_DEBUG
```

The valid log levels for this parameter are:

- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG

lsf.conf

- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

Run **badadmin reconfig** make changes take effect.

Default

Not defined (logging level is controlled by LSB_LOG_MASK).

LSB_MAIL_FROM_DOMAIN

Syntax

LSB_MAIL_FROM_DOMAIN=*domain_name*

Description

Windows only.

LSF uses the username as the from address to send mail. In some environments the from address requires domain information. If **LSB_MAIL_FROM_DOMAIN** is set, the domain name specified in this parameter will be added to the from address.

For example, if **LSB_MAIL_FROM_DOMAIN** is not set the, from address is SYSTEM; if LSB_MAIL_FROM_DOMAIN=example.com, the from address is SYSTEM@example.com.

Default

Not defined.

LSB_MAIL_TIMESTAMP_EPOCH_SECONDS

Syntax

LSB_MAIL_TIMESTAMP_EPOCH_SECONDS=y|Y|n|N

Description

Changes the time stamp format in LSF job mail. When the value is "Y" or "y", the epoch second time is used. By default, the conventional time format string is used.

Default

Not configured, the value is N.

LSB_MAILPROG

Syntax

LSB_MAILPROG=*file_name*

Description

Path and file name of the mail program used by LSF to send email. This is the electronic mail program that LSF uses to send system messages to the user. When LSF needs to send email to users it invokes the program defined by LSB_MAILPROG in `lsf.conf`.

In a mixed cluster, you can specify different programs for Windows and UNIX. You can set this parameter during installation on Windows.

LSF provides the **sendmail** program, which supports the **sendmail** protocol on UNIX. You can also write your own custom mail program and set `LSB_MAILPROG` to the path where this program is stored.

LSF also provides the `lsmail` program for Unix (supporting SMTP) and the `lsmail.exe` program for Windows (supporting both SMTP and Microsoft Exchange Server). If **lsmail** is specified, the parameter **LSB_MAILSERVER** must also be specified. On Windows, `lsmail.exe` can be configured directly. On Unix, the full path to the `lsmail` binary is required for configuration.

If you change your mail program, the LSF administrator must restart `sbatchd` on all hosts to retrieve the new value.

UNIX

By default, LSF uses **`/usr/lib/sendmail`** to send email to users. LSF calls **`LSB_MAILPROG`** with three arguments in the following order:

1. The `-oi` option: When reading a message from standard input, does not treat a line with only a period character (.) as the end of input. The end of the message is marked by an end-of-file character. If your mail program does not support this feature, skip this option.
2. The `-F 'LSF'` option specifies the full name of the sender.
3. The `-f` option specifies the return address for email.

For example, LSF calls the **`LSB_MAILPROG`** mail program with the following arguments: `mail_program -oi -F 'LSF' -f LSB_MAILSENDER_value <email_recipient>`

`LSB_MAILPROG` must read the body of the mail message from the standard input. Any program or shell script that accepts the arguments and input, and delivers the mail correctly, can be used.

`LSB_MAILPROG` must be executable by any user.

Windows

If `LSB_MAILPROG` is not defined, no email is sent.

Examples

```
LSB_MAILPROG=lsmail.exe
```

```
LSB_MAILPROG=/serverA/tools/lsf/bin/unixhost.exe
```

Default

Unix: **`/usr/lib/sendmail`**

Windows: empty

See also

`LSB_MAILSERVER`, **`LSB_MAILTO`**

LSB_MAILSENDER

Syntax

```
LSB_MAILSENDER=user_name
```

Description

Changes the default mail sender for job finish notifications.

By default, the user who submits a job receives an email notification when the job finishes. The default finish notification mail sender is the LSF admin.

However, there is a risk that the job submitter's mail box is full. If the job submitter's mail box is full, the notification email is returned to the mail sender (that is, the LSF admin). It is possible that the LSF admin will get a mail box full of returned notification mails, greatly inconveniencing the admin. Setting this parameter with a different user name for job finish notifications allows the admin's mail box to remain clear of returned notification emails. For example, you may set up a dedicated user from which all notification emails will be sent and this user will receive all returned notifications.

Default

Not defined.

LSB_MAILSERVER**Syntax**

For Windows: `LSB_MAILSERVER=mail_protocol:hostname`

For UNIX: `LSB_MAILSERVER=hostname`

Description

Part of mail configuration on Windows and Unix.

This parameter only applies when **lsmail** is used as the mail program (`LSB_MAILPROG=lsmail.exe`). Otherwise, it is ignored.

On Windows, specify the protocol and name of the mail server host. For an SMTP mail host, specify `SMTP:hostname`. For an exchange mail host, specify `EXCHANGE:hostname`. That is:

`LSB_MAILSERVER=[SMTP: | EXCHANGE:]hostname`

On UNIX, specify just the name of the mail server host. That is:

`LSB_MAILSERVER=hostname`

Note: On Unix, only SMTP is supported as the *mail_protocol*.

This parameter is set during installation of LSF or is modified by the LSF administrator.

If this parameter is modified, the LSF administrator must restart sbatchd on all hosts to retrieve the new value.

Default

Not defined

See also

LSB_LOCALDIR, **LSB_MAILTO**, and **LSB_MAILPROG**

LSB_MAILSIZE_LIMIT**Syntax**

`LSB_MAILSIZE_LIMIT=email_size_KB`

Description

Limits the size in KB of the email containing job output information.

The system sends job information such as CPU, process and memory usage, job output, and errors in email to the submitting user account. Some batch jobs can create large amounts of output. To prevent large job output files from interfering with your mail system, use `LSB_MAILSIZE_LIMIT` to set the maximum size in KB of the email containing the job information. Specify a positive integer.

If the size of the job output email exceeds `LSB_MAILSIZE_LIMIT`, the output is saved to a file under `JOB_SPOOL_DIR` or to the default job output directory if `JOB_SPOOL_DIR` is not defined. The email informs users of where the job output is located.

If the `-o` option of **bsub** is used, the size of the job output is not checked against `LSB_MAILSIZE_LIMIT`.

If you use a custom mail program specified by the `LSB_MAILPROG` parameter that can use the `LSB_MAILSIZE` environment variable, it is not necessary to configure `LSB_MAILSIZE_LIMIT`.

Default

By default, `LSB_MAILSIZE_LIMIT` is not enabled. No limit is set on size of batch job output email.

See also

LSB_MAILPROG, **LSB_MAILTO**

LSB_MAILTO

Syntax

`LSB_MAILTO=mail_account`

Description

LSF sends electronic mail to users when their jobs complete or have errors, and to the LSF administrator in the case of critical errors in the LSF system. The default is to send mail to the user who submitted the job, on the host on which the daemon is running; this assumes that your electronic mail system forwards messages to a central mailbox.

The **LSB_MAILTO** parameter changes the mailing address used by LSF. **LSB_MAILTO** is a format string that is used to build the mailing address.

Common formats are:

- `!U`: Mail is sent to the submitting user's account name on the local host. The substring `!U`, if found, is replaced with the user's account name.
- `!U@company_name.com`: Mail is sent to `user@company_name.com` on the mail server. The mail server is specified by the **LSB_MAILSERVER** parameter.
- `!U@!H`: Mail is sent to `user@submission_hostname`. The substring `!H` is replaced with the name of the submission host. This format is valid on UNIX only. It is not supported on Windows.

All other characters (including any other `!` characters) are copied exactly.

If this parameter is modified, the LSF administrator must restart `sbatchd` on all hosts to retrieve the new value.

Windows only: When a job exception occurs (for example, a job is overrun or underrun), an email is sent to the primary administrator set in the `lsf.cluster.cluster_name` file to the domain set in **LSB_MAILTO**. For example, if the primary administrator is `lsfadmin` and `LSB_MAILTO=fred@company.com`, an email is sent to `lsfadmin@company.com`. The email must be a valid Windows email account.

Default

`!U`

See also

LSB_MAILPROG, **LSB_MAILSIZE_LIMIT**

LSB_MAX_ASKED_HOSTS_NUMBER

Syntax

LSB_MAX_ASKED_HOSTS_NUMBER=*integer*

Description

Limits the number of hosts a user can specify with the `-m` (host preference) option of the following commands:

- **bsub**
- **brun**
- **bmod**
- **brestart**
- **brsvadd**
- **brsvmod**
- **brsvs**

The job is rejected if more hosts are specified than the value of **LSB_MAX_ASKED_HOSTS_NUMBER**.



CAUTION:

If this value is set high, there will be a performance effect if users submit or modify jobs using the `-m` option and specify a large number of hosts. 512 hosts is the suggested upper limit.

Valid values

Any whole, positive integer.

Default

512

LSB_MAX_FORWARD_PER_SESSION

Syntax

LSB_MAX_FORWARD_PER_SESSION=*integer*

Description

MutliCluster job forwarding model only. Sets the maximum number of jobs forwarded within a scheduling session.

Defined in the submission cluster only.

Default

50

LSB_MAX_JOB_DISPATCH_PER_SESSION

Sets the maximum number of job decisions that **mbschd** can make during one job scheduling session.

Syntax

LSB_MAX_JOB_DISPATCH_PER_SESSION=*integer*

Description

The system sets **LSB_MAX_JOB_DISPATCH_PER_SESSION** automatically during **mbatchd** startup, but you can adjust it manually:

Both **mbatchd** and **sbatchd** must be restarted when you manually change the value of this parameter.

Default

LSB_MAX_JOB_DISPATCH_PER_SESSION = Min (MAX(300, Total CPUs), 3000)

See also

MAX_SBD_CONNS and **JOB_DISPATCH_PACK_SIZE** in `lsb.params` and **LSF_NON_PRIVILEGED_PORTS** in `lsf.conf`

LSB_MAX_PACK_JOBS**Syntax**

LSB_MAX_PACK_JOBS=*integer*

Description

Applies to job packs only. Enables the job packs feature and specifies the maximum number of job submission requests in one job pack.

If the value is 0, job packs are disabled.

If the value is 1, jobs from the file are submitted individually, as if submitted directly using the `bsub` command.

We recommend 100 as the initial pack size. Tune this parameter based on cluster performance. The larger the pack size, the faster the job submission rate is for all the job requests the job submission file. However, while **mbatchd** is processing a pack, **mbatchd** is blocked from processing other requests, so increasing pack size can affect **mbatchd** response time for other job submissions.

If you change the configuration of this parameter, you must restart **mbatchd**.

Parameters related to job packs are not supported as environment variables.

Valid Values

Any positive integer or 0.

Default

Set to 300 at time of installation for the `HIGH_THROUGHPUT` configuration template. If otherwise undefined, then 0 (disabled).

LSB_MAX_PROBE_SBD**Syntax**

LSB_MAX_PROBE_SBD=*integer*

Description

Specifies the maximum number of **sbatchd** instances that can be polled by **mbatchd** in the interval **MBD_SLEEP_TIME/10**. Use this parameter in large clusters to reduce the time it takes for **mbatchd** to probe all **sbatchd**s.

The value of **LSB_MAX_PROBE_SBD** cannot be greater than the number of hosts in the cluster. If it is, **mbatchd** adjusts the value of **LSB_MAX_PROBE_SBD** to be same as the number of hosts.

After modifying **LSB_MAX_PROBE_SBD**, use **badadmin mbdrestart** to restart **mbatchd** and let the modified value take effect.

If **LSB_MAX_PROBE_SBD** is defined, the value of **MAX_SBD_FAIL** in **lsb.params** can be less than 3.

After modifying **LSB_MAX_PROBE_SBD**, use **badadmin mbdrestart** to restart **mbatchd** and let the modified value take effect.

Valid values

Any number between 0 and 64.

Default

20

See also

MAX_SBD_FAIL in **lsb.params**

LSB_MBD_BUSY_MSG**Syntax**

```
LSB_MBD_BUSY_MSG="message_string"
```

Description

Specifies the message displayed when **mbatchd** is too busy to accept new connections or respond to client requests.

Define this parameter if you want to customize the message.

Valid values

String, either non-empty or empty.

Default

Not defined. By default, LSF displays the message "LSF is processing your request. Please wait..."

Batch commands retry the connection to **mbatchd** at the intervals specified by the parameters **LSB_API_CONNTIMEOUT** and **LSB_API_RECVTIMEOUT**.

LSB_MBD_CONNECT_FAIL_MSG**Syntax**

```
LSB_MBD_CONNECT_FAIL_MSG="message_string"
```

Description

Specifies the message displayed when internal system connections to **mbatchd** fail.

Define this parameter if you want to customize the message.

Valid values

String, either non-empty or empty.

Default

Not defined. By default, LSF displays the message "Cannot connect to LSF. Please wait..."

Batch commands retry the connection to mbatchd at the intervals specified by the parameters LSB_API_CONNTIMEOUT and LSB_API_RECVTIMEOUT.

LSB_MBD_DOWN_MSG**Syntax**

LSB_MBD_DOWN_MSG="*message_string*"

Description

Specifies the message displayed by the **bhosts** command when mbatchd is down or there is no process listening at either the LSB_MBD_PORT or the LSB_QUERY_PORT.

Define this parameter if you want to customize the message.

Valid values

String, either non-empty or empty.

Default

Not defined. By default, LSF displays the message "LSF is down. Please wait..."

Batch commands retry the connection to mbatchd at the intervals specified by the parameters LSB_API_CONNTIMEOUT and LSB_API_RECVTIMEOUT.

LSB_MBD_PORT

See LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT.

LSB_MBD_QUERY_RETRY_MSG**Syntax**

LSB_MBD_QUERY_RETRY_MSG="*message_string*"

Description

Specifies the message displayed by the **bjobs** command when the number of concurrent **bjobs** queries exceeds the value of the **MAX_CONCURRENT_JOB_QUERY** parameter specified in the lsb.params file.

Define this parameter if you want to customize the message.

Valid values

String, either non-empty or empty.

Default

Not defined. By default, LSF displays the message "Batch system concurrent job query limit exceeded. LSF is retrying. Please wait..."

LSB_MC_CHKPNT_RERUN**Syntax**

LSB_MC_CHKPNT_RERUN=y | n

Description

For checkpointable MultiCluster jobs, if a restart attempt fails, the job is rerun from the beginning (instead of from the last checkpoint) without administrator or user intervention.

The submission cluster does not need to forward the job again. The execution cluster reports the job's new pending status back to the submission cluster, and the job is dispatched to the same host to restart from the beginning

Default

n

LSB_MC_DISABLE_HOST_LOOKUP**Syntax**

LSB_MC_DISABLE_HOST_LOOKUP=Y

Description

Disable submit host name lookup for remote jobs. When this parameter is set, the job **sbatchd** does not look up the submission host name when executing or cleaning a remote job. LSF will not be able to do any host dependent automounting.

Default

N. LSF will look up submit host name for remote jobs.

LSB_MC_INITFAIL_MAIL**Syntax**

LSB_MC_INITFAIL_MAIL=Y | All | Administrator

Description

MultiCluster job forwarding model only.

Specify Y to make LSF email the job owner when a job is suspended after reaching the retry threshold.

Specify Administrator to make LSF email the primary administrator when a job is suspended after reaching the retry threshold.

Specify All to make LSF email both the job owner and the primary administrator when a job is suspended after reaching the retry threshold.

Default

not defined

LSB_MC_INITFAIL_RETRY**Syntax**

LSB_MC_INITFAIL_RETRY=*integer*

Description

MultiCluster job forwarding model only. Defines the retry threshold and causes LSF to suspend a job that repeatedly fails to start, unless a forwarded job could not start in remote cluster due to insufficient

resources. For example, specify 2 retry attempts to make LSF attempt to start a job 3 times before suspending it.

Default

5

LSB_MC_RECALL_EXIT_INFO

Syntax

LSB_MC_RECALL_EXIT_INFO=Y | y | N | n

Description

For LSF multicluster capability, enables TERM_MC_RECALL to be a job termination reason (exitInfo) to distinguish recalled jobs from other exited jobs in the execution cluster..

While balancing workload between multiple clusters when using the LSF multicluster capability, jobs are recalled to the submission cluster because either the **MAX_RSCHED_TIME** is reached or **brequeue** is run on the submission cluster. If this parameter is defined as Y, these recalled jobs show the job termination reason (exitInfo) as TERM_MC_RECALL for these recalled jobs, otherwise these jobs show as exited jobs from the execution cluster.

Jobs that are rescheduled after running **brequeue** -a or -p are not considered as recalled jobs because these jobs are not returned back to the submission cluster for rescheduling.

To hide these recalled jobs from the list of exited jobs when running **bjobs**, **bhist**, or **bacct** on the execution cluster, define LSB_MC_RECALLED_JOBS_HIDE=Y in the `lsf.conf` file for the execution cluster.

Default

N

See also

LSB_MC_RECALLED_JOBS_HIDE

LSB_MC_RECALLED_JOBS_HIDE

Syntax

LSB_MC_RECALLED_JOBS_HIDE=Y | y | N | n

Description

For LSF multicluster capability, hides recalled jobs from the list of all jobs.

While balancing workload between multiple clusters when using the LSF multicluster capability, jobs are recalled to the submission cluster because either the **MAX_RSCHED_TIME** is reached or **brequeue** is run on the submission cluster. If LSB_MC_RECALL_EXIT_INFO=Y is defined in `lsf.conf`, these recalled jobs show the job termination reason (exitInfo) as TERM_MC_RECALL for these recalled jobs, otherwise these jobs show as exited jobs from the execution cluster.

If LSB_MC_RECALLED_JOBS_HIDE=Y is defined in the `lsf.conf` file for the execution cluster, LSF hides these recalled jobs from the list of exited jobs when running **bjobs**, **bhist**, or **bacct** on the execution cluster.

Default

N

See also

LSB_MC_RECALL_EXIT_INFO

LSB_MEMLIMIT_ENFORCE

Syntax

LSB_MEMLIMIT_ENFORCE=y | n

Description

Specify y to enable LSF memory limit enforcement.

If enabled, LSF sends a signal to kill all processes that exceed queue-level memory limits set by MEMLIMIT in `lsb.queues` or job-level memory limits specified by **bsub -M mem_limit**.

Otherwise, LSF passes memory limit enforcement to the OS. UNIX operating systems that support RLIMIT_RSS for **setrlimit()** can apply the memory limit to each process.

The following operating systems do not support memory limit at the OS level:

- Windows
- Sun Solaris 2.x

Default

Not defined. LSF passes memory limit enforcement to the OS.

See also

`lsb.queues`

LSB_MEMLIMIT_ENF_CONTROL

Syntax

LSB_MEMLIMIT_ENF_CONTROL=<Memory Threshold>:<Swap Threshold>:<Check Interval>:
[all]

Description

This parameter further refines the behavior of enforcing a job memory limit. In the case that one or more jobs reach a specified memory limit (both the host memory and swap utilization has reached a configurable threshold) at execution time, the worst offending job will be killed. A job is selected as the worst offending job on that host if it has the most overuse of memory (actual memory usage minus memory limit of the job).

You also have the choice of killing all jobs exceeding the thresholds (not just the worst).

The following describes usage and restrictions on this parameter.

- <Memory Threshold>: (Used memory size/maximum memory size)
A threshold indicating the maximum limit for the ratio of used memory size to maximum memory size on the host.
The threshold represents a percentage and must be an integer between 1 and 100.
- <Swap Threshold>: (Used swap size/maximum swap size)
A threshold indicating the maximum limit for the ratio of used swap memory size to maximum swap memory size on the host.

The threshold represents a percentage and must be an integer between 0 and 100. If you specify a value of 0, the swap threshold is excluded from memory limit enforcement (that is, there is no limit for swap memory).

- `<Check Interval>`: The value, in seconds, specifying the length of time that the host memory and swap memory usage will not be checked during the nearest two checking cycles.

The value must be an integer greater than or equal to the value of **SBD_SLEEP_TIME**.

- The keyword `:all` can be used to terminate all single host jobs that exceed the memory limit when the host threshold is reached. If not used, only the worst offending job is killed.
- If the cgroup memory enforcement feature is enabled (**LSB_RESOURCE_ENFORCE** includes the keyword "memory"), **LSB_MEMLIMIT_ENF_CONTROL** is ignored.
- The host will be considered to reach the threshold when both Memory Threshold and Swap Threshold are reached.
- **LSB_MEMLIMIT_ENF_CONTROL** does not have any effect on jobs running across multiple hosts. They will be terminated if they are over the memory limit regardless of usage on the execution host.
- On some operating systems, when the used memory equals the total memory, the OS may kill some processes. In this case, the job exceeding the memory limit may be killed by the OS not an LSF memory enforcement policy.

In this case, the exit reason of the job will indicate "killed by external signal".

Default

Not enabled. All jobs exceeding the memory limit will be terminated.

LSB_MIG2PEND

Syntax

```
LSB_MIG2PEND=0 | 1
```

Description

Applies only to migrating checkpointable or rerunnable jobs.

When defined with a value of 1, LSF requeues migrating jobs instead of restarting or rerunning them on the first available host. LSF requeues the jobs in the PEND state in order of the original submission time and with the original job priority.

If you want to place the migrated jobs at the bottom of the queue without considering submission time, define both **LSB_MIG2PEND=1** and **LSB_REQUEUE_TO_BOTTOM=1** in `lsf.conf`.

Ignored in a MultiCluster environment.

Default

Not defined. LSF restarts or reruns migrating jobs on the first available host.

See also

LSB_REQUEUE_TO_BOTTOM

LSB_MIXED_PATH_DELIMITER

Syntax

```
LSB_MIXED_PATH_DELIMITER=" | "
```

Description

Defines the delimiter between UNIX and Windows paths if **LSB_MIXED_PATH_ENABLE=y**. For example, /home/tmp/J.out|c:\tmp\J.out.

Default

A pipe "|" is the default delimiter.

See also

LSB_MIXED_PATH_ENABLE

LSB_MIXED_PATH_ENABLE**Syntax**

LSB_MIXED_PATH_ENABLE=y | n

Description

Allows you to specify both a UNIX and Windows path when submitting a job in a mixed cluster (both Windows and UNIX hosts).

The format is always unix_path_cmd|windows_path_cmd.

Applies to the following options of **bsub**:

- -o, -oo
- -e, -eo
- -i, -is
- -cwd
- -E, -Ep
- CMD
- queue level PRE_EXEC, POST_EXEC
- application level PRE_EXEC, POST_EXEC

For example:

```
bsub -o "/home/tmp/job%J.out|c:\tmp\job%J.out" -e "/home/tmp/err%J.out|c:\tmp\err%J.out" -E "sleep 9| sleep 8" -Ep "sleep 7| sleep 6" -cwd "/home/tmp|c:\tmp" "sleep 121|sleep 122"
```

The delimiter is configurable: **LSB_MIXED_PATH_DELIMITER**.

Note:

LSB_MIXED_PATH_ENABLE doesn't support interactive mode (**bsub -I**).

Default

Not defined. LSF jobs submitted .

See also

LSB_MIXED_PATH_DELIMITER

LSB_MOD_ALL_JOBS

Syntax

LSB_MOD_ALL_JOBS=y | Y

Description

If set, enables **bmod** to modify resource limits and location of job output files for running jobs.

After a job has been dispatched, the following modifications can be made:

- CPU limit (-c [*hour:*]*minute*[/*host_name* | /*host_model*] | -cn)
- Memory limit (-M *mem_limit* | -Mn)
- Rerunnable jobs (-r | -rn)
- Resource requirements (-R "*res_req*" except -R "cu[*cu_string*]")
- Run limit (-W *run_limit*[/*host_name* | /*host_model*] | -Wn)
- Standard output file name (-o *output_file* | -on)
- Standard error file name (-e *error_file* | -en)
- Overwrite standard output (stdout) file name up to 4094 characters for UNIX or 255 characters for Windows (-oo *output_file*)
- Overwrite standard error (stderr) file name up to 4094 characters for UNIX or 255 characters for Windows (-eo *error_file*)

To modify the CPU limit or the memory limit of running jobs, the parameters LSB_JOB_CPULIMIT=Y and LSB_JOB_MEMLIMIT=Y must be defined in `lsf.conf`.

Important:

Always run **badmin mbdrestart** after modifying LSB_MOD_ALL_JOBS.

Default

Set to Y at time of installation. If otherwise undefined, then N.

See also

LSB_JOB_CPULIMIT, LSB_JOB_MEMLIMIT

LSB_NCPU_ENFORCE

Description

When set to 1, enables parallel fairshare and considers the number of CPUs when calculating dynamic priority for queue-level user-based fairshare. LSB_NCPU_ENFORCE does not apply to host-partition user-based fairshare. For host-partition user-based fairshare, the number of CPUs is automatically considered.

Default

Not defined.

LSB_NQS_PORT

Syntax

LSB_NQS_PORT=*port_number*

Description

Required for LSF to work with NQS.

TCP service port to use for communication with NQS.

Where defined

This parameter can alternatively be set as an environment variable or in the services database such as `/etc/services`.

Example

```
LSB_NQS_PORT=607
```

Default

Not defined

LSB_NUM_NIOS_CALLBACK_THREADS**Syntax**

```
LSB_NUM_NIOS_CALLBACK_THREADS=integer
```

Description

Specifies the number of callback threads to use for batch queries.

If your cluster runs a large amount of blocking mode (**bsub -K**) and interactive (**bsub -I**) jobs, response to batch queries can become very slow. If you run a large number of **bsub -I** or **bsub -K** jobs, you can define the threads to the number of processors on the master host.

Default

Not defined

LSB_PACK_MESUB**Syntax**

```
LSB_PACK_MESUB=Y|y|N|n
```

Description

Applies to job packs only.

If `LSB_PACK_MESUB=N`, `mesub` will not be executed for any jobs in the job submission file, even if there are esubs configured at the application level (-a option of **bsub**), or using `LSB_ESUB_METHOD` in `lsf.conf`, or through a named esub executable under `LSF_SERVERDIR`.

If `LSB_PACK_MESUB=Y`, `mesub` is executed for every job in the job submission file.

Parameters related to job packs are not supported as environment variables.

Default

Y

LSB_PACK_SKIP_ERROR**Syntax**

LSB_PACK_SKIP_ERROR=Y|y|N|n

Description

Applies to job packs only.

If LSB_PACK_SKIP_ERROR=Y, all requests in the job submission file are submitted, even if some of the job submissions fail. The job submission process always continues to the end of the file.

If LSB_PACK_SKIP_ERROR=N, job submission stops if one job submission fails. The remaining requests in the job submission file are not submitted.

If you change the configuration of this parameter, you must restart mbatchd.

Parameters related to job packs are not supported as environment variables.

Default

N

LSB_PERF_METRICS_LOGDIR**Syntax**

LSB_PERF_METRICS_LOGDIR=/tmp

Description

Sets the directory in which mbatchd performance metric data is logged. The primary owner of this directory is the LSF administrator.

Default

LSF_LOGDIR

LSB_PERF_METRICS_SAMPLE_PERIOD**Syntax**

LSB_PERF_METRICS_SAMPLE_PERIOD=minutes

Description

Determines the sampling period for which mbatchd performance metric data is collected. The sampling period should not be too long, such as days.

Default

5 minutes

LSB_PLAN_KEEP_RESERVE

Specifies whether to keep the resource reservations for jobs with plans, even if the plan is no longer valid, until the next round of job plans are ready.

Syntax

LSB_PLAN_KEEP_RESERVE=Y|y|N|n

Description

Enabling this parameter prevents LSF from releasing resource reservations if the job plan becomes invalid (for example, because the host is down). If LSF releases the resource reservations, LSF might dispatch other jobs before the jobs with plans, even if the jobs with plans have a higher priority.

If **LSB_PLAN_KEEP_RESERVE** is set to Y | y, LSF keeps resource reservations for jobs with plans, even if the plan is no longer valid, until LSF creates new plans based on updated resource availability.

If **LSB_PLAN_KEEP_RESERVE** is set to N | n, LSF releases the resource reservations for jobs with plans that are no longer valid.

Default

N.

LSB_POSTEXEC_SEND_MAIL**Syntax**

LSB_POSTEXEC_SEND_MAIL=Y|y/N/n

Description

Enable this parameter to have LSF send an email to the user that provides the details of post execution, if any. This includes any applicable output.

Default

N

LSB_PROFILE_MBD**Syntax**

LSB_PROFILE_MBD=Y | N | "*keyword*[*value*] ..."

Description

This parameter configures the **mbatchd** daemon profiler to track the time that **mbatchd** spends on key functions.

If this parameter is set to Y, the profiler dumps **mbatchd** profiling information using the default settings.

Use the following keywords to configure the **mbatchd** profiler:

INTERVAL[*seconds*]

Specifies the sampling interval, which is how often the profiler dumps the profile data. Specify an integer greater than 0. The default is 300 seconds.

THRESHOLD[*threshold_portion*]

Specifies a threshold value that represents the portion of the last sampling interval. Only function stacks with a wall clock time value that exceeds this value are reported in the profile dump. Specify a float value between 0 and 1. The default is 0.01.

LOG_LEVEL[*log_level*]

Specifies the level at which to log the profile. Specify one of the standard log levels. The default is LOG_INFO.

TIME_LEVEL[*time_level*]

Specifies a timing level for checking how long the profiler runs, in milliseconds. Specify an integer greater than 0. The default is 5.

Default

N

If set to Y, the **mbatchd** profiler runs with the following settings:

```
LSB_PROFILE_MBD="INTERVAL[300] THRESHOLD[0.01] LOG_LEVEL[LOG_INFO] TIME_LEVEL[5]"
```

See also

LSB_PROFILE_SCH

LSB_PROFILE_SCH**Syntax**

```
LSB_PROFILE_SCH=Y | N | "keyword[value] ..."
```

Description

This parameter configures the **mbschd** daemon profiler to track the time that **mbschd** spends on key functions.

If this parameter is set to Y, the profiler dumps **mbschd** profiling information using the default settings.

Use the following keywords to configure the **mbschd** profiler:

INTERVAL[seconds]

Specifies the sampling interval, which is how often the profiler dumps the profile data. Specify an integer greater than 0. The default is 300 seconds.

THRESHOLD[threshold_portion]

Specifies a threshold value that represents the portion of the last sampling interval. Only function stacks with a wall clock time value that exceeds this value are reported in the profile dump. Specify a float value between 0 and 1. The default is 0.01.

LOG_LEVEL[log_level]

Specifies the level at which to log the profile. Specify one of the standard log levels. The default is LOG_INFO.

TIME_LEVEL[time_level]

Specifies a timing level for checking how long the profiler runs, in milliseconds. Specify an integer greater than 0. The default is 5.

Default

N

If set to Y, the **mbschd** profiler runs with the following settings:

```
LSB_PROFILE_SCH="INTERVAL[300] THRESHOLD[0.01] LOG_LEVEL[LOG_INFO] TIME_LEVEL[5]"
```

See also

LSB_PROFILE_MBD

LSB_PROJECT_LIMITS**Syntax**

```
LSB_PROJECT_LIMITS=Y|y|N|n
```

Description

If this parameter is set to Y|y, LSF uses **bconf** to create project-based limits when the cluster starts up. This might reduce the job scheduling cycle and improve job scheduling performance.

If there are already project limits defined when the cluster starts up, LSF ignores this parameter.

Default

N. LSF decides the best configuration for job scheduling performance.

LSB_QUERY_ENH

LSB_QUERY_ENH extends multithreaded query support to several common batch query commands (in addition to **bjobs** query requests).

Syntax

LSB_QUERY_ENH=Y|N

Description

When **LSB_QUERY_ENH** is enabled, multithreaded query support is added to most of the common batch query commands. If **LSB_QUERY_ENH** is not enabled, multithreaded query support is only supported for the **bjobs** command. The following batch query commands do not support multithreaded batch queries

The **mbatchd** system query monitoring mechanism starts automatically instead of waiting to be triggered by a query request. This ensures a faster and more consistent query response time within the system.

After you change this parameter

After you change this parameter, you must run **badmin mbdrestart -p -C** to restart **mbatchd**.

Default

N (multithreaded query support for **bjobs** query requests only)

Y. Running the upgrade installation also sets this parameter to Y.

LSB_QUERY_PORT

Specifies the port number that the **mbatchd** daemon uses for LSF query requests.

Syntax

LSB_QUERY_PORT=*port_number*

Description

This parameter must be defined on all clusters if you are using the LSF multicluster capability.

Use this parameter for busy clusters with many jobs and frequent query requests to increase **mbatchd** daemon performance when you use the **bjobs** command.

The *port_number* is the TCP/IP port number to be used by **mbatchd** to only service query requests from the LSF system. The **mbatchd** daemon checks the query port during initialization.

If the **LSB_QUERY_PORT** parameter *is not* defined:

- The **mbatchd** daemon uses the port that is specified by the **LSB_MBD_PORT** parameter in the `lsf.conf` file. If **LSB_MBD_PORT** is not defined, the **mbatchd** daemon looks into the system services database for port numbers to communicate with other hosts in the cluster.
- For each query request that the **mbatchd** daemon receives, it forks one child **mbatchd** daemon to service the request. Each child **mbatchd** daemon processes one request and then exits.

If the **LSB_QUERY_PORT** parameter is defined:

- The **mbatchd** daemon prepares this port for connection. The default behavior of **mbatchd** changes, a child **mbatchd** is forked, and the child **mbatchd** creates threads to process requests.
The interval that is used by the **mbatchd** daemon for forking new child **mbatchd** daemons is specified by the **MBD_REFRESH_TIME** parameter in the `lsb.params` file.
- The child **mbatchd** daemon takes over and listens on the port to process more query requests. For each request, the child **mbatchd** daemon creates a thread to process it.

After you define the **LSB_QUERY_PORT** parameter you can also define the following parameters:

- The **MBD_REFRESH_TIME** parameter in the `lsb.params` file specifies the time interval when the **mbatchd** daemon will fork a new child **mbatchd** daemon to service query requests.
- The **NEWJOB_REFRESH** parameter in the `lsb.params` enables a child **mbatchd** daemon to get up-to-date information about new jobs from the parent **mbatchd** daemon. If you enable this parameter, you must set the **MBD_REFRESH_TIME** parameter to a value greater than 10 seconds.

Restrictions

Applies only to UNIX operating systems that support thread programming.

Default

Not defined

Set to 6891 at time of installation. Otherwise, not defined.

See also

MBD_REFRESH_TIME and **NEWJOB_REFRESH** in `lsb.params`

LSB_QUERY_RETRY_INTERVAL_STEP

Syntax

`LSB_QUERY_RETRY_INTERVAL_STEP=seconds`

Description

Defines how often LSF libraries attempt to contact **mbatchd** or perform a concurrent jobs query. For each subsequent attempt to contact **mbatchd** or perform a concurrent jobs query, LSF increases the interval by this amount.

For example, if `LSB_QUERY_RETRY_INTERVAL_STEP=2` and the initial attempt fails, LSF makes the first retry attempt two seconds after the initial attempt, and if this attempt fails, LSF makes the second retry four seconds after the last retry attempt, and continues until the attempt succeeds or the **LSB_NTRIES** value is reached.

Default

1 second

LSB_RC_DEFAULT_HOST_TYPE

LSF resource connector default host type.

Syntax

`LSB_RC_DEFAULT_HOST_TYPE=string`

Description

Specifies the default host type to use for a template if the `type` attribute is not defined on a template in the template configuration files.

Example

```
LSB_RC_DEFAULT_HOST_TYPE=X86_64
```

Default

X86_64

LSB_RC_EXTERNAL_HOST_FLAG

Setting the **LSB_RC_EXTERNAL_HOST_FLAG** parameter enables the LSF resource connector feature.

Syntax

```
LSB_RC_EXTERNAL_HOST_FLAG="string ..."
```

Description

Specify a list of Boolean resource names that identify host providers that are available for borrowing. Any hosts or instances that provide a resource from the list are initially closed by LSF at startup. Hosts and instances are only opened when the resource connector informs LSF that the host was successfully allocated or the instance is launched.

Run the **badmin mbdrestart** command for this parameter to take effect.

Example

```
LSB_RC_EXTERNAL_HOST_FLAG="awshost googlehost azurehost"
```

Default

Not defined

LSB_RC_EXTERNAL_HOST_IDLE_TIME

For LSF resource connector. If no jobs are running on a resource provider instance for the specified number of minutes, LSF relinquishes the instances.

Syntax

```
LSB_RC_EXTERNAL_HOST_IDLE_TIME=minutes
```

Description

If the **LSB_RC_EXTERNAL_HOST_IDLE_TIME** parameter is set to 0, the policy is disabled and the resource provider instance is never shut down for lack of jobs.

Example

```
LSB_RC_EXTERNAL_HOST_IDLE_TIME=30
```

Default

60 minutes

LSB_RC_EXTERNAL_HOST_MAX_TTL

For LSF resource connector. Maximum time-to-live for a resource provider instance. If an instance is in the cluster for this number of minutes, LSF closes it and its status goes to `closed_RC`).

Syntax

`LSB_RC_EXTERNAL_HOST_MAX_TTL=minutes`

Description

If the resource provider is still active after the specified number of minutes, LSF changes the host status to `closed_RC`, which prevents the host from accepting additional workload. If the `LSB_RC_EXTERNAL_HOST_MAX_TTL` parameter is set to 0, the policy is disabled. The cloud resource is returned and terminated if the workload that associated with the host is done.

You cannot use the **admin hopen** command to open a borrowed host in `closed_RC` status.

Example

```
LSB_RC_EXTERNAL_HOST_MAX_TTL=30
```

Default

0 minutes (disabled)

LSB_RC_MQTT_ERROR_LIMIT

For LSF resource connector. The maximum number of API error messages that are stored in Mosquitto per host provider.

Syntax

`LSB_RC_MQTT_ERROR_LIMIT=integer`

Description

Run **admin mbdrestart** for any change to take effect.

Example

```
LSB_RC_MQTT_ERROR_LIMIT=20
```

Default

10

LSB_RC_QUERY_INTERVAL

For LSF resource connector. The interval in seconds that the resource connector checks host status and asynchronous requests from a resource provider.

Syntax

`LSB_RC_QUERY_INTERVAL=seconds`

Description

Run **admin mbdrestart** for any change to take effect.

Example

```
LSB_RC_QUERY_INTERVAL=60
```

Default

30 seconds

LSB_RC_TEMPLATE_REQUEST_DELAY

For LSF resource connector. The amount of time that LSF waits before repeating a request for a template, in minutes, if the **ebrokerd** daemon encountered certain provider errors in a previous request.

Syntax

```
LSB_RC_TEMPLATE_REQUEST_DELAY=integer
```

Description

This parameter takes effect if the **ebrokerd** daemon encounters one of the following provider errors when requesting a template from the host provider API:

- **InsufficientAddressCapacity**: Not enough available addresses to satisfy the minimum request.
- **InsufficientCapacity**: Not enough capacity to satisfy the import instance request.
- **InsufficientInstanceCapacity**: Not enough instance capacity available to satisfy the request.
- **InsufficientHostCapacity**: Not enough capacity to satisfy the dedicated host request.
- **InsufficientReservedInstanceCapacity**: Not enough available reserved instances to satisfy the minimum request.

Run **badmin mbdrestart** for any change to take effect.

Example

```
LSB_RC_TEMPLATE_REQUEST_DELAY=20
```

Default

10

LSB_RC_UPDATE_INTERVAL

For LSF resource connector. Configures how often LSF calculates demand for pending jobs and publishes this demand to the **ebrokerd** daemon.

Syntax

```
LSB_RC_UPDATE_INTERVAL=seconds
```

Description

This parameter updates the demand calculation according to the specified interval instead of calculating demand every scheduler cycle to avoid performance impact.

Example

```
LSB_RC_UPDATE_INTERVAL=20
```

Default

30 seconds

LSB_REQUEUE_TO_BOTTOM**Syntax**

LSB_REQUEUE_TO_BOTTOM=0 | 1

Description

Specify 1 to put automatically requeued jobs at the bottom of the queue instead of at the top. Also requeues migrating jobs to the bottom of the queue if **LSB_MIG2PEND** is also defined with a value of 1.

Specify 0 to requeue jobs to the top of the queue.

Ignored in a MultiCluster environment.

Default

0 (LSF requeues jobs to the top of the queue).

See also

LSB_MIG2PEND, **REQUEUE_EXIT_VALUES** in `lsb.queues`

LSB_RESOURCE_ENFORCE**Syntax**

```
LSB_RESOURCE_ENFORCE="resource [resource]"
```

Description

Controls resource enforcement through the Linux cgroup memory, cpuset subsystem, and GPU access on Linux systems with cgroup support. Memory and cpuset enforcement for Linux cgroups is supported on Red Hat Enterprise Linux (RHEL) 6.2 or above, SuSe Linux Enterprise Linux 11 SP2 or above. GPU enforcement for Linux cgroup device subsystems is supported on Red Hat Enterprise Linux 6.2 or above, and SuSe Linux Enterprise Linux 11 SP2 or above.

resource can be `memory`, `cpu`, or `gpu`. You may also specify two or three of these resources in any order.

LSF can impose strict host-level memory and swap limits on systems that support Linux cgroups. These limits cannot be exceeded. All LSF job processes are controlled by the Linux cgroup system. If job processes on a host use more memory than the defined limit, the job will be immediately killed by the Linux cgroup memory subsystem. Memory is enforced on a per job/per host basis, not per task. If the host OS is Red Hat Enterprise Linux 6.3 or above, cgroup memory limits are enforced, and LSF is notified to terminate the job. Additional notification is provided to users through specific termination reasons displayed by **bhist -l**.

To enable memory enforcement, configure **LSB_RESOURCE_ENFORCE="memory"**.

Note:

- If **LSB_RESOURCE_ENFORCE="memory"** is configured, all existing LSF memory limit related parameters such as **LSF_HPC_EXTENSIONS="TASK_MEMLIMIT"**, **LSF_HPC_EXTENSIONS="TASK_SWAPLIMIT"**, **LSB_JOB_MEMLIMIT** and **LSB_MEMLIMIT_ENFORCE** are ignored.
- **LSB_RESOURCE_ENFORCE="memory"** must be specified for host-based memory and swap limit enforcement with the **bsub -hl** command option to take effect. If no memory or swap limit is specified for the job (the merged limit for the job, queue, and application profile, if specified), or

LSB_RESOURCE_ENFORCE="memory" is not specified, a host-based memory limit is not set for the job. When **LSB_RESOURCE_ENFORCE="memory"** is configured and memory and swap limits are specified for the job, but `-h1` is *not* specified, memory and swap limits are calculated and enforced as a multiple of the number of tasks running on the execution host.

LSF can also enforce CPU affinity binding on systems that support the Linux cgroup cpuset subsystem. When CPU affinity binding through Linux cgroups is enabled, LSF will create a cpuset to contain job processes if the job has affinity resource requirements, so that the job processes cannot escape from the allocated CPUs. Each affinity job cpuset includes only the CPU and memory nodes that LSF distributes. Linux cgroup cpusets are only created for affinity jobs.

To enable CPU enforcement, configure **LSB_RESOURCE_ENFORCE="cpu"**.

If you are enabling memory and CPU enforcement through the Linux cgroup memory cpuset subsystems after upgrading an existing LSF cluster, make sure that the following parameters are set in `lsf.conf`:

- **LSF_PROCESS_TRACKING=Y**
- **LSF_LINUX_CGROUP_ACCT=Y**

To enable GPU access through Linux cgroups, configure **LSB_RESOURCE_ENFORCE="gpu"**. LSF creates devices to contain job processes if the job has GPU resource requirements so that the job processes cannot escape from the allocated GPUs. Each GPU job device includes only the GPUs that LSF distributes. Linux cgroup devices are only created for GPU jobs.

Note: When GPU enforcement is enabled, the GPUs that are contained in one device cgroup are reallocated new GPU IDs, beginning with 0. CUDA Version 7.0 or newer supports cgroup completely.

Jobs can specify how job processes are to be bound to these computing elements. LSF uses the environment variable `CUDA_VISIBLE_DEVICES` to tell user applications which GPUs are allocated. It is possible for user applications to escape from the allocated GPUs by changing `CUDA_VISIBLE_DEVICES` directly to use other GPUs.

For example, the following command submits a job with one exclusive thread GPU requirement:

```
bsub -R "rusage[ngpus_excl_t=1]" ./myapp
```

LSF creates a device that contains one exclusive thread GPU and attaches the process ID of the application `./myapp` to this device. The device serves as a strict container for job processes, so that the application `./myapp` cannot use other GPUs.

Examples

For a parallel job with 3 tasks and a memory limit of 100 MB, such as the following:

```
bsub -n 3 -M 100 -R "span[ptile=2]" blaunch ./mem_eater
```

The application `mem_eater` keeps increasing the memory usage. LSF will kill the job if it consumes more than 200 MB total memory on one host. For example, if `hosta` runs 2 tasks and `hostb` runs 1 task, the job will only be killed if total memory on exceeds 200 MB on either `hosta` or `hostb`. If one of the tasks consumes more than 100 MB memory but less than 200 MB, and the other task doesn't consume any memory, the job will not be killed. That is, LSF does not support per task memory enforcement for cgroups.

For a job with affinity requirement, such as the following:

```
bsub -R "affinity[core:membind=localonly]" ./myapp
```

LSF will create a cpuset which contains one core and attach the process ID of the application `./myapp` to this cpuset. The cpuset serves as a strict container for job processes, so that the application `./myapp` cannot bind to other CPUs. LSF will add all memory nodes into the cpuset to make sure the job can access all memory nodes on the host, and will make sure job processes will access preferred memory nodes first.

Default

Not defined. Resource enforcement through the Linux cgroup system is not enabled.

LSB_RLA_PORT**Syntax**

LSB_RLA_PORT $port_number$

Description

TCP port used for communication between the LSF topology adapter (RLA) and the HPC scheduler plugin.

Default

6883

LSB_RLA_UPDATE**Syntax**

LSB_RLA_UPDATE= $time_seconds$

Description

Specifies how often the HPC scheduler refreshes free node information from the LSF topology adapter (RLA).

Default

600 seconds

LSB_RLA_WORKDIR**Syntax**

LSB_RLA_WORKDIR= $directory$

Description

Directory to store the LSF topology adapter (RLA) status file. Allows RLA to recover its original state when it restarts. When RLA first starts, it creates the directory defined by LSB_RLA_WORKDIR if it does not exist, then creates subdirectories for each host.

You should avoid using `/tmp` or any other directory that is automatically cleaned up by the system. Unless your installation has restrictions on the LSB_SHARED_DIR directory, you should use the default for LSB_RLA_WORKDIR.

Default

LSB_SHARED_DIR/ $cluster_name$ /rla_workdir

LSB_SACCT_ONE_UG**Syntax**

LSB_SACCT_ONE_UG= $y | Y | n | N$

Description

When set to Y, minimizes overall memory usage of **mbatchd** during fairshare accounting at job submission by limiting the number of share account nodes created on **mbatchd** startup. Most useful when there are a lot of user groups with **all** members in the fairshare policy.

When a default user group is defined, inactive user share accounts are still defined for the default user group.

When setting this parameter, you must restart the **mbatchd**.

Default

N

LSB_SBD_PORT

See LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT.

LSB_SBD_STARTUP_RETRY_INTERVAL

Syntax

LSB_SBD_STARTUP_RETRY_INTERVAL=*time_seconds*

Description

If **sbatchd** failed to initialize socket, this parameter specifies the amount of time before **sbatchd** tries again (if **LSB_SBD_STARTUP_RETRY_TIMES** is specified).

Valid values

Any positive integer between 1 and **INFINIT_INT**.

INFINIT_INT is defined in `lsf.h`.

Default

If **LSB_SBD_STARTUP_RETRY_TIMES** is defined, the default value is 1.

Otherwise, not defined.

See also

LSB_SBD_STARTUP_RETRY_TIMES

LSB_SBD_STARTUP_RETRY_TIMES

Syntax

LSB_SBD_STARTUP_RETRY_TIMES=*retry_limit*

Description

The maximum number of times for **sbatchd** to try to initialize a socket when the first attempt failed.

Valid values

Any integer between 0 and **INFINIT_INT**.

INFINIT_INT is defined in `lsf.h`.

Default

0. **sbatchd** does not try to initialize a socket if the first attempt failed.

See also

LSB_SBD_STARTUP_RETRY_INTERVAL

LSB_SET_TMPDIR**Syntax**

```
LSB_SET_TMPDIR=y | n | <ENV_VAR_NAME>
```

If *y*, LSF sets the `TMPDIR` environment variable, overwriting the current value with the job-specific temporary directory. For more details on the job-specific temporary directory, refer to **LSB_JOB_TMPDIR**.

If this parameter is set to the name of an environment variable (for example, **MY_TMPDIR**), LSF sets the value of this environment variable to the job-specific temporary directory. The user application can use this environment variable within the code.

Example

```
LSB_SET_TMPDIR=MY_TMPDIR
```

On Unix, the name of this environment variable is `$MY_TMPDIR` and its value is the job-specific temporary directory.

On Windows, the name of this environment variable is `%MY_TMPDIR%` and its value is the job-specific temporary directory.

Default

n

LSB_SHAREDIR**Syntax**

```
LSB_SHAREDIR=directory
```

Description

Directory in which the job history and accounting logs are kept for each cluster. These files are necessary for correct operation of the system. Like the organization under **LSB_CONFDIR**, there is one subdirectory for each cluster.

The **LSB_SHAREDIR** directory must be owned by the LSF administrator. It must be accessible from all hosts that can potentially become the master host, and must allow read and write access from the master host.

The `LSB_SHAREDIR` directory typically resides on a reliable file server.

Default

```
LSF_INDEP/work
```

See also

LSB_LOCALDIR

LSB_SHORT_HOSTLIST

Syntax

```
LSB_SHORT_HOSTLIST=1
```

Description

Displays an abbreviated list of hosts in **bjobs** and **bhist** for a parallel job where multiple processes of a job are running on a host. Multiple processes are displayed in the following format:

```
processes*hostA
```

For example, if a parallel job is running 5 processes on hostA, the information is displayed in the following manner:

```
5*hostA
```

Setting this parameter may improve mbatchd restart performance and accelerate event replay.

Default

Set to 1 at time of installation for the HIGH_THROUGHPUT and PARALLEL configuration templates. Otherwise, not defined.

LSB_SIGSTOP

Syntax

```
LSB_SIGSTOP=signal_name | signal_value
```

Description

Specifies the signal sent by the SUSPEND action in LSF. You can specify a signal name or a number.

If this parameter is not defined, by default the SUSPEND action in LSF sends the following signals to a job:

- Parallel or interactive jobs: SIGTSTP is sent to allow user programs to catch the signal and clean up. The parallel job launcher also catches the signal and stops the entire job (task by task for parallel jobs). Once LSF sends SIGTSTP, LSF assumes the job is stopped.
- Other jobs: SIGSTOP is sent. SIGSTOP cannot be caught by user programs. The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the **kill -l** command.

Example

```
LSB_SIGSTOP=SIGKILL
```

In this example, the SUSPEND action sends the three default signals sent by the TERMINATE action (SIGINT, SIGTERM, and SIGKILL) 10 seconds apart.

Default

Not defined. Default SUSPEND action in LSF is sent.

LSB_SKIP_FULL_HOSTS

Syntax

```
LSB_SKIP_FULL_HOSTS=y|Y|n|N
```

Description

Specifies whether LSF removes full hosts (that is, hosts with all slots occupied) from consideration at the beginning of each scheduling session.

When selecting hosts to run jobs, LSF removes unusable hosts from consideration at the beginning of each scheduling session. Hosts that are unusable by any job include hosts that are down (`unavail` or `unreach`), closed by the administrator (`closed_Adm`), or closed due to a load threshold (`closed_Busy`), and removing these hosts early in the scheduling session helps with scheduling performance.

However, LSF does not remove full hosts (`closed_Full`) from consideration because these hosts can still be used by jobs in preemptive queues if queue-based preemption is enabled.

If **LSB_SKIP_FULL_HOSTS** is enabled (set to Y), LSF removes full hosts from consideration at the beginning of each scheduling session, as long as there is no preemption relationship between the queues (for more details, refer to the **PREEMPTION** parameter in the `lsb_queues` file).

Default

N

LSB_SSH_XFORWARD_CMD

For X11 forwarding, you can modify the default value with an SSH command (full PATH and options allowed).

Syntax

`LSB_SSH_XFORWARD_CMD=[/path[/path]]ssh command [ssh options]`

Description

Optional when submitting jobs with SSH X11 forwarding. Allows you to specify an SSH command and options when a job is submitted with **-XF**.

Replace the default value with an SSH command (full PATH and options allowed).

When running a job with the **-XF** option, runs the SSH command specified here.

Default

`ssh -X -n`

LSF_STAGE_IN_EXEC

Specifies the stage in script for direct data staging (for example, IBM CAST burst buffer).

Syntax

`LSF_STAGE_IN_EXEC=script_name`

Description

This script is run as root during the stage in process. The specified script must be located in the `$LSF_SERVERDIR` directory.

Default

Not defined.

LSB_STAGE_MAX_STAGE_IN

Specifies the maximum number of concurrent stage in processes that can run on the host.

Syntax

LSB_STAGE_MAX_STAGE_IN=*num_processes*

Description

If the number of concurrent stage-in processes reaches this limit, LSF keeps the plan for this job, but will not trigger a stage-in job. This can occur for direct data staging jobs where the job handles the file transfer instead of LSF because LSF does not control the stage-in operations.

Default

Not defined.

LSB_STAGE_OUT_EXEC

Specifies the stage out script for direct data staging (for example, IBM CAST burst buffer).

Syntax

LSB_STAGE_OUT_EXEC=*script_name*

Description

This script is run as root during the stage out process. The specified script must be located in the \$LSF_SERVERDIR directory.

Default

Not defined.

LSB_STAGE_STORAGE

Specifies the resource name to report available storage space for direct data staging (for example, IBM CAST burst buffer).

Syntax

LSB_STAGE_STORAGE=*resource_name* [*total_resource_name*]

Description

Optionally, specify the resource name to report the total storage space for direct data staging as the second resource name. This prevents LSF from assigning more storage than is available because the resource information is out of date. This can occur for direct data staging jobs where the job handles the file transfer instead of LSF because LSF cannot reliably predict the storage usage for these jobs.

If both resource names are specified, the resource to report the available storage must be configured as a dynamic, decreasing, and numeric, while the resource to report the maximum storage must be configured as numeric.

If only one resource name is specified, the resource must be configured as a dynamic, decreasing resource. LSF assumes that the total storage space on the host is the sum of the reported available storage space, the reserved space, and the used space. The reserved space is storage space that is assigned to jobs that have a plan, but have not completed the stage in operation. The used space is storage space that is assigned to jobs that completed the stage in operation, but has not completed the stage out operation.

Examples

```
LSB_STAGE_STORAGE=ssd
```

The `ssd` resource reports the available storage space. LSF estimates that the total storage space available is the sum of the available storage space, the storage space that is reserved for planned jobs, and the storage space that is used by jobs that completed the stage in operation (but not the stage out operation).

```
LSB_STAGE_STORAGE=ssd ssd_max
```

The `ssd` resource reports the available storage space. The `ssd_max` resource reports the total storage space.

Default

Not defined.

LSB_STAGE_TRANSFER_RATE

Specifies the estimated data transfer rate for direct data staging (for example, IBM CAST burst buffer). LSF uses this value to calculate the predicted duration for data stage in.

Syntax

```
LSB_STAGE_TRANSFER_RATE=data_transfer_rate
```

Description

Specify the estimated data transfer rate in kilobits per second (kbps).

Default

1 gbps (1048576 kbps).

LSB_STEP_CGROUP_DEFAULT

Specifies the default value for the `bsub -step_cgroup` option for CSM jobs.

Syntax

```
LSB_STEP_CGROUP_DEFAULT=y | n
```

Description

Run the `bsub -step_cgroup` option at job submission time to override this default value.

Default

Not defined.

LSB_START_EBROKERD

Enables the `mbatchd` daemon to start the `ebrokerd` daemon whenever `mbatchd` starts up, is reconfigured, or when it detects that the old `ebrokerd` daemon exits.

Syntax

```
LSB_START_EBROKERD = Y | y | N | n
```

Description

When the **LSB_START_EBROKERD** parameter is set to Y or y , the **mbatchd** daemon starts the **ebrokerd** daemon whenever **mbatchd** starts up, is reconfigured, or when it detects that the old **ebrokerd** daemon exists. This setting is required to use advance reservation pre-scripts and post-scripts.

If this parameter is set to N or n, **ebrokerd** is started only for host power control and the LSF resource connector.

Default

N

LSB_START_MPS

LSF starts CUDA MPS (Multi-Process Service) for the GPU jobs that require only GPUs with **EXCLUSIVE_PROCESS** or **DEFAULT** modes.

Syntax

LSB_START_MPS=Y|y|N|n

Description

CUDA MPS allows multiple CUDA processes to share a single GPU context with these modes. If the job requires GPUs with **EXCLUSIVE_THREAD** mode, LSF does not start CUDA MPS for the GPU jobs.

Remember: When you change the value of the **LSB_START_MPS** parameter, you must restart the **sbatchd** daemon:

```
badmin hrestart all
```

When LSF starts MPS for a job, LSF sets **CUDA_MPS_PIPE_DIRECTORY** instead of **CUDA_VISIBLE_DEVICES**. The GPU jobs communicate with MPS through a named pipe that is defined by **CUDA_MPS_PIPE_DIRECTORY**. **CUDA_MPS_PIPE_DIRECTORY** is stored under the directory that is specified by **LSF_TMPDIR**. When the job finishes, LSF removes the pipe.

If the cgroup feature is enabled, LSF also creates a cgroup for MPS under the job level cgroup.

The MPS Server supports up to 16 client CUDA contexts concurrently. This limitation is per user per job and means that MPS can only support up to 16 CUDA processes at one time even if LSF allocated multiple GPUs. MPS cannot exit normally if GPU jobs are killed. The LSF cgroup feature can help resolve this situation.

The MPS function is supported by CUDA Version 5.5, or later.

The **LSB_START_JOB_MPS** environment variable at the job level overrides the **LSB_START_MPS** parameter.

When the **LSB_GPU_NEW_SYNTAX=Y** parameter is set, the default value of the MPS option is `mps=no` in the new GPU requirement syntax. This option overrides the parameters **LSB_START_MPS = y** in the `lsf.conf` file and the **LSB_START_JOB_MPS=y** environment variable, and `bsub -env LSB_START_JOB_MPS=y` command option.

Default

N - LSF does not start CUDA MPS for GPU jobs.

LSB_STDOUT_DIRECT**Syntax**

LSB_STDOUT_DIRECT=y|Y|n|N

Description

When set, and used with the `-o` or `-e` options of **bsub**, redirects standard output or standard error from the job directly to a file as the job runs.

If **LSB_STDOUT_DIRECT** is not set and you use the **bsub -o** option, the standard output of a job (as well as the resource usage summary) is written to a temporary file and copied to the file you specify *after* the job finishes.

LSB_STDOUT_DIRECT is not supported on Windows.

Default

Not defined

LSB_STOP_IGNORE_IT**Usage**

LSB_STOP_IGNORE_IT= Y | y

Description

Allows a solitary job to be stopped regardless of the idle time (IT) of the host that the job is running on. By default, if only one job is running on a host, the host idle time must be zero in order to stop the job.

Default

Not defined

LSB_SUB_COMMANDNAME**Syntax**

LSB_SUB_COMMANDNAME=y | Y

Description

If set, enables **esub** to use the variable **LSB_SUB_COMMAND_LINE** in the **esub** job parameter file specified by the **LSB_SUB_PARM_FILE** environment variable.

The **LSB_SUB_COMMAND_LINE** variable carries the value of the **bsub** command argument, and is used when **esub** runs.

Example

esub contains:

```
#!/bin/sh
. $LSB_SUB_PARM_FILE

exec 1>&2
if [ $LSB_SUB_COMMAND_LINE="netscape" ];

then
    echo "netscape is not allowed to run in batch mode"
    exit $LSB_SUB_ABORT_VALUE
fi
```

LSB_SUB_COMMAND_LINE is defined in **\$LSB_SUB_PARM_FILE** as:

```
LSB_SUB_COMMAND_LINE=netscape
```

A job submitted with:

lsf.conf

```
bsub netscape ...
```

Causes **esub** to echo the message:

```
netscape is not allowed to run in batch mode
```

Default

Not defined

See also

LSB_SUB_COMMAND_LINE and **LSB_SUB_PARM_FILE** environment variables

LSB_SUBK_SHOW_EXEC_HOST

Syntax

```
LSB_SUBK_SHOW_EXEC_HOST=Y|N
```

Description

When enabled, displays the execution host in the output of the command **bsub -K**. If the job runs on multiple hosts, only the first execution host is shown.

In a MultiCluster environment, this parameter must be set in both clusters.

Tip:

Restart **sbatchd** on the execution host to make changes take effect.

Default

Set to Y at time of installation. If otherwise undefined, then N.

LSB_SUPPRESS_CUSTOM_REASONS

Allows individual users to disable display of customized pending reasons for the Single key reason (**bjobs -p1**) and Categorized Pending Reasons (**bjobs -p2** and **bjobs -p3**).

Syntax

```
LSB_SUPPRESS_CUSTOM_REASONS=Y|y|N|n
```

Description

This parameter applies to all **bjobs -p** levels except **-p0**. The command **bjobs -p0**, is used to display pending reasons in the style prior to version 10.1, without using the single key reason or the categorized pending reason features.

Default

N (customized pending reason messages are enabled)

See also

LSB_BJOBS_PENDREASON_LEVEL in `lsf.conf`.

LSB_TERMINAL_SERVICE_PORT

Syntax

```
LSB_TERMINAL_SERVICE_PORT=port_number
```

Description

Terminal service port number for Remote Desktop Protocol (RDP). This port is used in **tssub** jobs.

Default

Not defined.

LSB_TIME_CMD**Syntax**

LSB_TIME_CMD=*timing_level*

Description

The timing level for checking how long batch commands run.

Time usage is logged in milliseconds; specify a positive integer.

Example: LSB_TIME_CMD=1

Default

Not defined

See also

LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_TIME_DMD

The timing level for checking how long the **dmd** daemon runs.

Syntax

LSB_TIME_DMD=*timing_level*

Description

Specify a positive integer. Time usage is logged in milliseconds.

```
LSB_TIME_DMD=1
```

You must run the **bdata admin reconfig** command to reconfigure the LSF data manager daemon (**dmd**) for the parameter to take effect.

Default

0

See also

LSB_TIME_CMD, LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_TIME_GPD**Syntax**

LSB_TIME_GPD=*timing_level*

Description

Sets a timing level for checking how long **gpolicyd** routines run. Time usage is logged in milliseconds. Specify a positive integer.

Default

Not defined.

LSB_TIME_MBD

Syntax

LSB_TIME_MBD=*timing_level*

Description

The timing level for checking how long mbatchd routines run. Time usage is logged in milliseconds; specify a positive integer. Example: LSB_TIME_MBD=1

Default

Not defined

See also

LSB_TIME_CMD, LSB_TIME_SBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_TIME_SCH

Syntax

LSB_TIME_SCH=*timing_level*

Description

The timing level for checking how long mbschd routines run. Time usage is logged in milliseconds; specify a positive integer. Example: LSB_TIME_SCH=1

Default

Not defined

LSB_TIME_RESERVE_NUMJOBS

Syntax

LSB_TIME_RESERVE_NUMJOBS=*maximum_reservation_jobs*

Description

Enables time-based slot reservation. The value must be positive integer. LSB_TIME_RESERVE_NUMJOBS controls maximum number of jobs using time-based slot reservation. For example, if LSB_TIME_RESERVE_NUMJOBS=4, only the top 4 jobs get their future allocation information.

Use `LSB_TIME_RESERVE_NUMJOBS=1` to allow only the highest priority job to get accurate start time prediction.

Recommended value

3 or 4 is the recommended setting. Larger values are not as useful because after the first pending job starts, the estimated start time of remaining jobs may be changed.

Default

Not defined

LSB_TIME_SBD

The timing level for checking how long `sbatchd` routines run.

Syntax

`LSB_TIME_SBD=timing_level`

Description

Specify a positive integer. Time usage is logged in milliseconds.

Example: `LSB_TIME_SBD=1`

Default

Not defined

See also

`LSB_TIME_CMD`, `LSB_TIME_DMD`, `LSB_TIME_MBD`, `LSF_TIME_LIM`, `LSF_TIME_RES`

LSB_TSJOBS_HELPER_HOSTS

Syntax

`LSB_TSJOBS_HELPER_HOSTS="helper_host_list"`

helper_host_list is a space-separated list of hosts that are Terminal Services job helper hosts.

Description

Lists the Terminal Services job helper hosts. Helper hosts must be LSF servers in the LSF cluster. Configure a maximum of 256 hosts in the list.

The local helper service will select one host from the list and send requests to the helper to create a user session. If the host fails to create the user session, the next helper host in the list will be tried. The local helper service will not select itself as helper host.

For stability, you should configure one helper host for every 40 execution hosts.

Example: `LSB_TSJOBS_HELPER_HOSTS="host1 host2 host3"`

To make the modified parameter take effect, restart the LIM (by running `lsadmin limrestart all`) and restart the **TSJobHelper** Windows service on the execution hosts.

Default

None

LSB_TSJOBS_HELPER_PORT

Syntax

LSB_TSJOBS_HELPER_PORT=*port_number*

Description

Specify the service port to use for communication with TSJobHelper.

For example: LSB_TSJOBS_HELPER_PORT=6889

TSJobHelper uses this port to communicate with the helper hosts.

To make the modified parameter take effect, restart the LIM (by running **lsadmin limrestart all**) and restart the **TSJobHelper** Windows service on the execution hosts and helper hosts.

Default

6889

LSB_TSJOBS_HELPER_TIMEOUT

Syntax

LSB_TSJOBS_HELPER_TIMEOUT=seconds

Description

Specify the maximum time out that the local TSJobHelper service waits for a helper host to reply. After time out, the local service tries the next helper host in the LSB_TSJOBS_HELPER_HOSTS list.

Example: LSB_TSJOBS_HELPER_TIMEOUT=60

To make the modified parameter take effect, restart the LIM (by running **lsadmin limrestart all**) and restart the **TSJobHelper** Windows service on the execution hosts.

Default

60 seconds

LSB_USE_SEQUENTIAL_SIGNAL

Syntax

LSB_USE_SEQUENTIAL_SIGNAL=*y|Y|n|N*

Description

When this parameter is set to Y or y, the transition sequence of signals for parallel jobs that are killed is the same as the sequence of signals for sequential jobs that are killed. That is, when this parameter is set to Y or y the signal sequence for parallel jobs that are killed is SIGINT, SIGTERM, then SIGKILL.

The minimum time between each signal transition is defined by the **JOB_TERMINATE_INTERVAL** parameter in lsb.params, which is 10 seconds by default.

Default

N. When killing a parallel job, only the **SIGKILL** signal is sent.

LSB_USER_REQUEUE_TO_BOTTOM

Syntax

LSB_USER_REQUEUE_TO_BOTTOM=1 | 0

Description

Determines whether jobs are requeued to the top or bottom of the queue.

When defined with a value of 1, LSF requeues jobs to the top of the queue. When defined with a value of 0, LSF requeues jobs to the bottom of the queue.

If you want to place the migrated jobs at the bottom of the queue without considering submission time, define both **LSB_MIG2PEND=1** and **LSB_REQUEUE_TO_BOTTOM=1** in `lsf.conf`.

Ignored in a MultiCluster environment.

Default

Not defined. LSF requeues jobs in order of original submission time and job priority.

See also

LSB_MIG2PEND

LSB_UTMP

Syntax

LSB_UTMP=y | Y

Description

If set, enables registration of user and account information for interactive batch jobs submitted with **bsub -Ip** or **bsub -Is**. To disable utmp file registration, set LSB_UTMP to any value other than y or Y; for example, `LSB_UTMP=N`.

LSF registers interactive batch jobs the job by adding a entries to the utmp file on the execution host when the job starts. After the job finishes, LSF removes the entries for the job from the utmp file.

Limitations

Registration of utmp file entries is supported on the following platforms:

- Solaris (all versions)
- HP-UX (all versions)
- Linux (all versions)

utmp file registration is not supported in a MultiCluster environment.

Because interactive batch jobs submitted with **bsub -I** are not associated with a pseudo-terminal, utmp file registration is not supported for these jobs.

Default

Not defined

LSF_AC_PNC_URL

Syntax

LSF_AC_PNC_URL=HTTP://*server_address*[:*port_number*]

Description

Specifies the URL and listen port of the LSF Application Center Notifications server for sending notifications. If the listen port is not specified, the default port number is 80.

Default

Not defined.

See also

bpost (1)

LSF_AM_OPTIONS

Syntax

LSF_AM_OPTIONS=AMFIRST | AMNEVER

Description

Determines the order of file path resolution when setting the user's home directory.

This variable is rarely used but sometimes LSF does not properly change the directory to the user's home directory when the user's home directory is automounted. Setting LSF_AM_OPTIONS forces LSF to change directory to \$HOME before attempting to automount the user's home.

When this parameter is not defined or set to AMFIRST, LSF, sets the user's \$HOME directory from the automount path. If it cannot do so, LSF sets the user's \$HOME directory from the passwd file.

When this parameter is set to AMNEVER, LSF, never uses automount to set the path to the user's home. LSF sets the user's \$HOME directory directly from the passwd file.

Valid values

The two values are AMFIRST and AMNEVER

Default

Same as AMFIRST

LSF_API_CONNTIMEOUT

Syntax

LSF_API_CONNTIMEOUT=*time_seconds*

Description

Timeout when connecting to LIM.

EGO parameter

EGO_LIM_CONNTIMEOUT

Default

5

See also**LSF_API_RECVMTIMEOUT****LSF_API_RECVMTIMEOUT****Syntax**LSF_API_RECVMTIMEOUT=*time_seconds***Description**

Timeout when receiving a reply from LIM.

EGO parameter

EGO_LIM_RECVMTIMEOUT

Default

20

See also**LSF_API_CONNTIMEOUT****LSF_ASPLUGIN****Syntax**LSF_ASPLUGIN=*path***Description**Points to the SGI Array Services library `libarray.so`. The parameter only takes effect on 64-bit x-86 Linux 2.6, glibc 2.3.**Default**`/usr/lib64/libarray.so`**LSF_AUTH****Syntax**LSF_AUTH=*eauth | ident | none***Description**

Specifies the external client to server authentication method that is used.. This parameter is required for any cluster that contains Windows hosts, and is optional for UNIX-only clusters. You must shut down the LSF daemons on all server hosts before defining or changing the value of **LSF_AUTH**, then restart the LSF daemons on all server hosts to apply the new authentication method.

eauth

For site-specific customized external client to server authentication. Provides the highest level of security of all LSF authentication methods.

ident

For client to server authentication using the RFC 931/1413/1414 protocol to verify the identity of the remote client. If you want to use ident authentication, you must download and install the ident protocol, available from the public domain, and register ident as required by your operating system.

none

Disables the triggering of additional authentication. For use in a trusted environment where more complex authentication methods are not required.

If `LSF_AUTH=none` is set, the value of **LSF_AUTH_DAEMONS** is ignored.

For UNIX-only clusters, privileged ports authentication (setuid) can be configured by commenting out or deleting the `LSF_AUTH` parameter. If you choose privileged ports authentication, LSF commands must be installed as setuid programs owned by root. If the commands are installed in an NFS-mounted shared file system, the file system must be mounted with setuid execution allowed, that is, without the `nosuid` option.

Restriction: To enable privileged ports authentication, **LSF_AUTH** must not be defined; setuid is not a valid value for **LSF_AUTH**.

Default

Set to `eauth` at time of installation. Otherwise, not defined.

During LSF installation, a default `eauth` executable is installed in the directory specified by the environment variable **LSF_SERVERDIR**. The default executable provides an example of how the `eauth` protocol works. You should write your own `eauth` executable to meet the security requirements of your cluster.

LSF_AUTH_DAEMONS**Syntax**

`LSF_AUTH_DAEMONS=y | Y`

Description

Enables LSF daemon authentication when external authentication is enabled (**LSF_AUTH=eauth** in the file `lsf.conf`). Daemons invoke **eauth** to authenticate each other as specified by the `eauth` executable.

Default

Not defined.

LSF_BIND_JOB

LSF_BIND_JOB Specifies the processor binding policy for sequential and parallel job processes that run on a single host.

Syntax

`LSF_BIND_JOB=NONE | BALANCE | PACK | ANY | USER | USER_CPU_LIST`

Description

Note: **LSF_BIND_JOB** is deprecated in LSF Standard Edition and LSF Advanced Edition. You should enable LSF CPU and memory affinity scheduling in with the `AFFINITY` parameter in `lsb.hosts`. If both `LSF_BIND_JOB` and affinity scheduling are enabled, affinity scheduling takes effect, and `LSF_BIND_JOB` is disabled. **LSF_BIND_JOB** and **BIND_JOB** are the only affinity options available in LSF Express Edition.

On Linux execution hosts that support this feature, job processes are hard bound to selected processors.

If processor binding feature is not configured with the **BIND_JOB** parameter in an application profile in `lsb.applications`, the **LSF_BIND_JOB** configuration setting `lsf.conf` takes effect. The application profile configuration for processor binding overrides the `lsf.conf` configuration.

For backwards compatibility:

- `LSF_BIND_JOB=Y` is interpreted as `LSF_BIND_JOB=BALANCE`
- `LSF_BIND_JOB=N` is interpreted as `LSF_BIND_JOB=NONE`

Supported platforms

Linux with kernel version 2.6 or higher

Default

Not defined. Processor binding is disabled.

LSF_BINDIR

Syntax

`LSF_BINDIR=directory`

Description

Directory in which all LSF user commands are installed.

Default

`LSF_MACHDEP/bin`

LSF_BMPLUGIN

Points to the bitmask library `libbitmask.so`.

Syntax

`LSF_BMPLUGIN=path`

Description

The parameter only takes effect on 64-bit x-86 Linux.

Default

`/usr/lib64/libbitmask.so`

LSF_CLEARCASE_WRAPPER_RETRY

LSF integration for IBM Rational ClearCase. Specifies the maximum number of times that the daemon wrapper tries to run the **cleartool setview** command.

Syntax

`LSF_CLEARCASE_WRAPPER_RETRY=retry_limit`

Description

In the LSF ClearCase daemon wrapper, the **cleartool setview** command creates a process that is set to a specified ClearCase view for **sbatchd** daemons. If the **setview** command fails, **sbatchd** cannot start and the job will fail or exit.

Together with the **LSF_CLEARCASE_WRAPPER_RETRY_INTERVAL** parameter, the **LSF_CLEARCASE_WRAPPER_RETRY** parameter enables a retry mechanism to resolve failures with the **cleartool setview** command so that LSF can tolerate temporary **cleartool setview** command failures.

For example, to allow one more attempt after the first attempt has failed, specify **LSF_CLEARCASE_WRAPPER_RETRY=1**. When you set the **LSF_CLEARCASE_WRAPPER_RETRY** parameter, you should also set a value for the **LSF_CLEARCASE_WRAPPER_RETRY_INTERVAL** parameter.

Default

Not defined. Daemon wrapper does not retry.

Enable

After you set or change the **LSF_CLEARCASE_WRAPPER_RETRY** parameter, use the **badadmin mbdrestart** and **badadmin hrestart all** commands to apply the change.

See also

LSF_CLEARCASE_WRAPPER_RETRY_INTERVAL

LSF_CLEARCASE_WRAPPER_RETRY_INTERVAL

LSF integration for IBM Rational ClearCase. Specifies the number of seconds to wait before the daemon wrapper tries to run the **cleartool setview** command.

Syntax

LSF_CLEARCASE_WRAPPER_RETRY_INTERVAL=*time_seconds*

Description

In the LSF ClearCase daemon wrapper, the **cleartool setview** command creates a process that is set to a specified ClearCase view for **sbatchd** daemons. If the **setview** command fails, **sbatchd** cannot start and the job will fail or exit.

Together with the **LSF_CLEARCASE_WRAPPER_RETRY** parameter, the **LSF_CLEARCASE_WRAPPER_RETRY_INTERVAL** parameter enables a retry mechanism to resolve failures with the **cleartool setview** command so that LSF can tolerate temporary **cleartool setview** command failures.

For example, to have the daemon wrapper wait 10 seconds before it retries the **cleartool setview** command, specify **LSF_CLEARCASE_WRAPPER_RETRY_INTERVAL=10**. The minimum wait time is 1 second.

For the **LSF_CLEARCASE_WRAPPER_RETRY_INTERVAL** parameter to take effect, you must also configure the **LSF_CLEARCASE_WRAPPER_RETRY** parameter.

Default

Not defined.

Enable

After you set or change the **LSF_CLEARCASE_WRAPPER_RETRY_INTERVAL** parameter, use the **badadmin mbdrestart** and **badadmin hrestart all** commands to apply the change.

See also

LSF_CLEARCASE_WRAPPER_RETRY

LSF_CMD_LOG_MASK

Syntax

LSF_CMD_LOG_MASK=*log_level*

Description

Specifies the logging level of error messages from LSF commands.

For example:

```
LSF_CMD_LOG_MASK=LOG_DEBUG
```

To specify the logging level of error messages, use LSB_CMD_LOG_MASK. To specify the logging level of error messages for LSF daemons, use LSF_LOG_MASK.

LSF commands log error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by LSF_CMD_LOG_MASK determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used for basic debugging. The level LOG_DEBUG3 records all debugging messages, and can cause log files to grow very large; it is not often used. Most debugging is done at the level LOG_DEBUG2.

The commands log to the `syslog` facility unless LSF_CMD_LOGDIR is set.

Valid values

The log levels from highest to lowest are:

- LOG_EMERG
- LOG_ALERT
- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

Default

LOG_WARNING

See also

LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD,
LSB_CMD_LOGDIR, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSF_CMD_LOGDIR

Syntax

LSF_CMD_LOGDIR=*path*

Description

The path to the log files used for debugging LSF commands.

This parameter can also be set from the command line.

Default

/tmp

See also

LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSF_COLLECT_ENERGY_USAGE**Syntax**

LSF_COLLECT_ENERGY_USAGE=N | Y

Description

Determines if the collection of job and node energy usage is enabled on the LSF cluster. This is used for CPU frequency management and energy usage reporting. The default value is N.

Default

N

LSF_CONF_RETRY_INT**Syntax**

LSF_CONF_RETRY_INT=*time_seconds*

Description

The number of seconds to wait between unsuccessful attempts at opening a configuration file (only valid for LIM). This allows LIM to tolerate temporary access failures.

EGO parameter

EGO_CONF_RETRY_INT

Default

30

See also

LSF_CONF_RETRY_MAX

LSF_CONF_RETRY_MAX**Syntax**

LSF_CONF_RETRY_MAX=*integer*

Description

The maximum number of retry attempts by LIM to open a configuration file. This allows LIM to tolerate temporary access failures. For example, to allow one more attempt after the first attempt has failed, specify a value of 1.

EGO parameter**EGO_CONF_RETRY_MAX****Default**

0

See also**LSF_CONF_RETRY_INT****LSF_CONFDIR****Syntax**LSF_CONFDIR=*directory***Description**

Directory in which all LSF configuration files are installed. These files are shared throughout the system and should be readable from any host. This directory can contain configuration files for more than one cluster.

The files in the **LSF_CONFDIR** directory must be owned by the primary LSF administrator, and readable by all LSF server hosts.

If live reconfiguration through the **bconf** command is enabled by the parameter **LSF_LIVE_CONFDIR**, configuration files are written to and read from the directory set by **LSF_LIVE_CONFDIR**.

Default

LSF_INDEP/conf

See also**LSB_CONFDIR, LSF_LIVE_CONFDIR****LSF_CONNECTION_CHANGE****Syntax**

LSF_CONNECTION_CHANGE=Y|y|N|n

Description

Windows only. Enables **lsreghost** to register with LSF servers whenever it detects a change in the total number of connections (IP addresses) that are associated with the local host.

This parameter is only valid if registration handling is enabled for LSF hosts (that is, LSF_REG_FLOAT_HOSTS=Y is set in the `lsf.conf` file on the LSF server).

Set this parameter to Y on the local host and **lsreghost** registers the local host with the LSF servers if it detects a change in the total number of connections (that is, IP addresses) that are associated with the local host.

lsf.conf

You must restart the **lsreghost** command on the local host for this change to take effect.

Default

N

See also

LSF_REG_FLOAT_HOSTS, LSF_REG_HOST_INTERVAL.

LSF_CPUSETLIB

Points to the SGI cpuset library `libcputset.so`.

Syntax

`LSF_CPUSETLIB=path`

Description

The parameter only takes effect on 64-bit x-86 Linux.

Default

`/usr/lib64/libcputset.so`

LSF_CRASH_LOG

Syntax

`LSF_CRASH_LOG=Y | N`

Description

On Linux hosts only, enables logging when or if a daemon crashes. Relies on the Linux debugger (gdb). Two log files are created, one for the root daemons (res, lim, sbd, and mbatchd) in `/tmp/lsf_root_daemons_crash.log` and one for administrative daemons (mbschd) in `/tmp/lsf_admin_daemons_crash.log`.

File permissions for both files are 600.

If enabling, you must restart the daemons for the change to take effect.

Default

N (no log files are created for daemon crashes)

LSF_CRAY_RELEASE_TIMEOUT

Syntax

`LSF_CRAY_RELEASE_TIMEOUT=time_seconds`

Description

Specifies the amount of time, in seconds, after the reservation on a Cray compute node is released before LSF allows the compute node to be used again in another reservation request.

Default

600 seconds.

LSF_CRAY_RUR_DIR

Syntax

LSF_CRAY_RUR_DIR=*file_path*

Description

Use this parameter to specify the file path to the RUR data files, which must be a shared file system that is accessible from any potential first execution host. The RUR data file for jobs that are submitted by all users is named `rur.output`. A job-specific RUR data file for a specific job ID is named `rur.<jobID>` (for example, for job ID 123, the file is named `rur.123`).

You can use the `%U` special character to represent the home directory of the user that submitted the job. For example, if you specify `LSF_CRAY_RUR_DIR=%U/.rur`, and userA and userB submitted jobs, the RUR data files are located in `/home/userA/.rur` for userA and `/home/userB/.rur` for userB.

Default

`LSB_SHAREDIR/cluster_name/craylinux/<cray_machine_name>/rur`

LSF_CRAY_RUR_EPILOG_PATH

Syntax

LSF_CRAY_RUR_EPILOG_PATH=*file_path_to_rur_epilogue.py*

Description

Use this parameter to specify the file path to the RUR epilog script file.

Note: LSF runs the epilog script file with the `-j jobID` option. Therefore, the epilog script file must support the `-j` option with the job ID as the argument.

Default

`opt/cray/rur/default/bin/rur_epilogue.py`

LSF_CRAY_RUR_PROLOG_PATH

Syntax

LSF_CRAY_RUR_PROLOG_PATH=*file_path_to_rur_prologue.py*

Description

Use this parameter to specify the file path to the RUR prolog script file.

Note: LSF runs the prolog script file with the `-j jobID` option. Therefore, the prolog script file must support the `-j` option with the job ID as the argument.

Default

`opt/cray/rur/default/bin/rur_prologue.py`

LSF_DATA_BSUB_CHKSUM

For jobs with a data requirement, this parameter enables the **bsub** and **bmod** commands to perform a full sanity check on the files and folders, and to generate the hash for each file and folder.

Syntax

LSF_DATA_BSUB_CHKSUM=Y | y | N | n

Description

For jobs with a data requirement, this parameter determines how LSF performs the following operations on the corresponding files or folders:

- Sanity check to see if the files or folders exist
- Check whether the submission user can access the files or folders
- Discover the size and modification times of the files or folders
- Generate a hash for each file or folder

If LSF_DATA_BSUB_CHKSUM is set to y or Y, the **bsub** and **bmod** commands perform all the operations on the files or folders.

If LSF_DATA_BSUB_CHKSUM is set to n or N, all the operations on the files or folders occur at the transfer job. This reduces the amount of time that the **bsub** or **bmod** commands take to submit or modify the job, which equalizes submission performance between jobs with and without data requirements.

You can force the **bsub** or **bmod** command to perform all operations on the files or folders by specifying the `-datachk` option with the **bsub** or **bmod** command. The `-datachk` option can be specified only with the **-data** command option.

Default

N. Full checking for the files or folders are performed at the transfer job and not by the **bsub** or **bmod** command.

LSF_DATA_CLUSTER

Syntax

LSF_DATA_CLUSTER=*string*

Description

The **LSF_DATA_CLUSTER** parameter should be used when multiple clusters use the same dmd in a non-multicluster environment. The clusters where the dmd does not live set the parameter to the cluster name where the dmd lives. The LSF Data Manager **bdata** command will check the `lsf.conf` configuration file for **LSF_DATA_CLUSTER** and use the configured value when invoking the `eauth` (unless a remote cluster is explicitly specified).

Example

LSF_DATA_CLUSTER=*my_data_manager_cluster_name*

Default

Not defined

LSF_DATA_HOSTS

Specifies a list of hosts where the LSF data manager daemon (**dmd**) can run, and where clients can contact the LSF data manager that is associated with the cluster. The **dmd** daemon can run only on the listed hosts.

Syntax

LSF_DATA_HOSTS=*host_list*

Description

All LSF data manager clients, including the **mbatchd** daemon, use this parameter to contact the **dmd** daemon. This parameter must be defined in every cluster that talks to the LSF data manager. Defining this parameter acts as a switch that enables the overall LSF data management features.

The order of host names in the list decides which host is the LSF data manager master host, and order of failover. All host names must be listed in the same order for all LSF data managers.

To have LIM start the LSF data manager daemon automatically, and keep monitoring it, the hosts that are listed in **LSF_DATA_HOSTS** must be LSF server hosts.

When the LSF data manager starts, it verifies that its current host name is on this list, and that it is an LSF server.

To change the **LSF_DATA_HOSTS** parameter, complete these steps:

1. Run the **lsadmin limshutdown** command to shut down LIM on the LSF data manager candidate hosts.
2. Run the **bdata admin shutdown** command to shut down the **dmd** daemon on the LSF data manager candidate hosts.
3. Change the parameter.
4. Run the **lsadmin limstartup** command to start LIM on the LSF data manager candidate hosts.
5. Run the **badmin mbdrestart** command to restart all LSF master candidate hosts.

Default

Not defined

LSF_DATA_NO_SSH_CHK_HOSTS

Specifies a list of data hosts for which **ssh** is not needed.

Syntax

LSF_DATA_NO_SSH_CHK_HOSTS="*host1 | regex1 ...*"

Description

For LSF Data Manager, if the host specified within the data specification of a submitted job matches one of the hosts in this parameter, LSF assumes that the submission host can directly access the file within the data specification. If the **LSF_DATA_BSUB_CHKSUM=N** is specified, LSF also assumes that all the transfer nodes can directly access the file in the data specification. LSF does not use **ssh** to process the file.

For LSF Data Manager, if the data source file is located in a host that is specified in the **LSF_DATA_NO_SSH_CHK_HOSTS** parameter, LSF Data Manager assumes that the transfer node can directly access the source file, and directly copies the source file to the staging area without using **ssh** to check the source file, otherwise the **bsub** command or transfer job might fail.

Default

Not defined

LSF_DATA_PORT

Port number of the LSF data manager daemon (**dmd**) associated with the cluster.

Syntax

LSF_DATA_PORT=*integer*

Description

After you change the **LSF_DATA_PORT** parameter, complete these steps:

1. Run the **lsadmin limshutdown** command to shut down LIM on the LSF data manager candidate hosts.
2. Run the **bdata admin shutdown** command to shut down the **dmd** daemon on the LSF data manager candidate hosts.
3. Change the parameter.
4. Run the **lsadmin limstartup** command to start LIM on the LSF data manager candidate hosts.
5. Run the **badmin mbdrestart** command to restart all LSF master candidate hosts.

Default

1729

LSF_DATA_SCP_CMD

Syntax

LSF_DATA_SCP_CMD=-q-o[PasswordAuthentication=no|yes] |[Batchmode=no|yes]

Description

In LSF, if the sshd daemon is configured to display a banner message to remote users, the unexpected output of the banner message might cause the use of ssh and scp within the LSF Data Manager to fail.

You can control the use of ssh and scp in LSF Data Manager by specifying two `lsf.conf` parameters **LSF_DATA_SSH_CMD** and **LSF_DATA_SCP_CMD**.

LSF_DATA_SSH_CMD defines the default SSH command.

LSF_DATA_SCP_CMD defines the default SCP command.

Note: The **LSF_DATA_SCP_CMD** definition does not apply to the job transfer script. To change the use of scp in the job transfer script, you must use the **FILE_TRANSFER_CMD** parameter in the `lsf.datamanager1` configuration file.

If you change the value of the **LSF_DATA_SSH_CMD** or **LSF_DATA_SCP_CMD** parameter, you must restart the dmd daemon to apply the changes by running the **bdata admin reconfig** command.

Valid Values

Valid options are as follows:

- -q
Quiet mode. Suppresses all warning and diagnostic messages.
- -o PasswordAuthentication=no
Specifies that password authentication is not allowed.
- -o BatchMode=yes
Specifies non-interactive mode (or batch mode). If password authentication is required, the command fails without prompting.

Default

The default value is "scp -q -o PasswordAuthentication=no -o BatchMode=yes -r -p"..

LSF_DATA_SKIP_GROUP_CHECK

When the **LSF_DATA_SKIP_GROUP_CHECK** parameter is set to Y or y, the LSF **mbatchd** daemon does not check whether the user who requests data is a member of the user group that is specified by the **-datagrpx** option for the job.

Syntax

LSF_DATA_SKIP_GROUP_CHECK=Y|y|N|n

Description

The **LSF_DATA_SKIP_GROUP_CHECK** parameter is ignored if the LSF data manager is not configured (no data hosts are configured with the **LSF_DATA_HOSTS** parameter and the **LSF_DATA_PORT** parameter is not configured).

After you change **LSF_DATA_SKIP_GROUP_CHECK**, run the **admin mbdrestart** command to restart all LSF master candidate hosts.

Default

N - if a data job specifies the **-datagrpx** option, LSF checks whether the user is a member of the group specified. If the user is not a member the job submission is rejected.

LSF_DATA_SSH_CMD**Syntax**

LSF_DATA_SSH_CMD=-q-o[PasswordAuthentication=no|yes] |[Batchmode=no|yes]

Description

In LSF, if the **sshd** daemon is configured to display a banner message to remote users, the unexpected output of the banner message might cause the use of **ssh** and **scp** within the LSF Data Manager to fail.

You can control the use of **ssh** and **scp** in LSF Data Manager by specifying two **lsf.conf** parameters **LSF_DATA_SSH_CMD** and **LSF_DATA_SCP_CMD**.

LSF_DATA_SSH_CMD defines the default SSH command.

LSF_DATA_SCP_CMD defines the default SCP command.

If you change the value of the **LSF_DATA_SSH_CMD** or **LSF_DATA_SCP_CMD** parameter, you must restart the **dmd** daemon to apply the changes by running the **bdata admin reconfig** command.

Valid Values

Valid options are as follows:

- -q

Quiet mode. Suppresses all warning and diagnostic messages.

- -o PasswordAuthentication=no

Specifies that password authentication is not allowed.

- -o BatchMode=yes

Specifies non-interactive mode (or batch mode). If password authentication is required, the command fails without prompting.

Default

The default value is "ssh -q -n -o PasswordAuthentication=no -o BatchMode=yes"..

LSF_DAEMON_WRAP**Syntax**

LSF_DAEMON_WRAP=y | Y

Description

When this parameter is set to y or Y, mbatchd, sbatchd, and RES run the executable daemons .wrap located in **LSF_SERVERDIR**.

This parameter is ignored in Kerberos integrations with LSF Versions 9.1.2, or newer.

Default

Not defined. LSF does not run the daemons .wrap executable.

LSF_DAEMONS_CPUS**Syntax**

LSF_DAEMONS_CPUS="*mbatchd_cpu_list:mbschd_cpu_list*"

mbatchd_cpu_list

Defines the list of master host CPUs where the mbatchd daemon processes can run (hard CPU affinity). Format the list as a white-space delimited list of CPU numbers.

mbschd_cpu_list

Defines the list of master host CPUs where the mbschd daemon processes can run. Format the list as a white-space delimited list of CPU numbers.

Description

By default, **mbatchd** and **mbschd** can run on any CPUs. If LSF_DAEMONS_CPUS is set, they only run on a specified list of CPUs. An empty list means LSF daemons can run on any CPUs. Use spaces to separate multiple CPUs.

The operating system can assign other processes to run on the same CPU; however, if utilization of the bound CPU is lower than utilization of the unbound CPUs.

Related parameters

To improve scheduling and dispatch performance of all LSF daemons, you should use LSF_DAEMONS_CPUS together with EGO_DAEMONS_CPUS (in ego.conf or lsf.conf), which controls LIM CPU allocation, and MBD_QUERY_CPUS, which binds **mbactchd** query processes to specific CPUs so that higher priority daemon processes can run more efficiently. To get best performance, CPU allocation for all four daemons should be assigned their own CPUs. For example, on a 4 CPU SMP host, the following configuration gives the best performance:

```
EGO_DAEMONS_CPUS=0 LSF_DAEMONS_CPUS=1:2 MBD_QUERY_CPUS=3
```

Examples

If you specify

```
LSF_DAEMONS_CPUS="1:2"
```

the `mbatchd` processes run only on CPU number 1 on the master host, and `mbschd` run on only on CPU number 2.

If you specify

```
LSF_DAEMONS_CPUS="1 2:1 2"
```

both `mbatchd` and `mbschd` run CPU 1 and CPU 2.

Important

You can specify CPU affinity only for master hosts that use one of the following operating systems:

- Linux 2.6 or higher
- Solaris 10 or higher

EGO parameter

`LSF_DAEMONS_CPUS=lim_cpu_list`: run the EGO LIM daemon on the specified CPUs.

Default

Not defined

See also

`MBD_QUERY_CPUS` in `lsb.params`

LSF_DCGM_PORT

Enables the NVIDIA Data Center GPU Manager (DCGM) features and specifies the port number that LSF uses to communicate with the DCGM daemon.

Syntax

`LSF_DCGM_PORT=port_number`

Description

Define this parameter to enable DCGM features with LSF. LSF uses this port to communicate with the DCGM daemon.

DCGM provides additional functionality when working with jobs that request GPU resources by:

- providing GPU usage information for **EXCLUSIVE_PROCESS** mode jobs.
- checking the status of GPUs to automatically filter out unhealthy GPUs when the job allocates GPU resources, and to automatically add back the GPU if it becomes healthy again. DCGM provides mechanisms to check the GPU health and LSF integrates these mechanisms to check the GPU status before, during, and after the job is running to meet the GPU requirements. If LSF detects that a GPU is not healthy before the job is complete, LSF requeues the job. This ensures that the job runs on healthy GPUs. If the execution host's DCGM status is not valid, the `bjobs -l` command shows an error message. The job still runs, but GPU resource usage reports are not available from that host.
- synchronizing the GPU auto-boost feature to support jobs that run across multiple GPUs, including jobs that run across multiple GPUs on a single host.

Run the `-gpu` option with the `bjobs`, `bhist`, and `bacct` commands to display GPU usage information from DCGM after the job finishes. The `-gpu` option must be used with the following command options:

- For the `bjobs` command, you must run the `-gpu` option with the `-l` or `-UF` options.
- For the `bhist` command, you must run the `-gpu` option with the `-l` option.
- For the `bacct` command, you must run the `-gpu` option with the `-l` option.

After changing this parameter, restart the **sbatchd** and **RES** daemons to apply the change.

After enabling the parameter, you must start up DCGM to use the features.

Note: If the DCGM integration does not work as expected due to a missing `libdcgm.so` file, create a softlink to ensure that the `libdcgm.so` file exists and is accessible:

```
sudo ln -s /usr/lib64/libdcgm.so.1 /usr/lib64/libdcgm.so
```

Default

Not defined. DCGM features are disabled.

See also

MBD_REFRESH_TIME and **NEWJOB_REFRESH** in `lsb.params`

LSF_DEBUG_CMD

Syntax

```
LSF_DEBUG_CMD=log_class
```

Description

Sets the debugging log class for LSF commands and APIs.

Specifies the log class filtering to be applied to LSF commands or the API. Only messages belonging to the specified log class are recorded.

LSF_DEBUG_CMD sets the log class and is used in combination with LSF_CMD_LOG_MASK, which sets the log level. For example:

```
LSF_CMD_LOG_MASK=LOG_DEBUG LSF_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Debugging is turned on when you define both parameters.

The daemons log to the `syslog` facility unless LSF_CMD_LOGDIR is defined.

To specify multiple log classes, use a space-separated list enclosed by quotation marks. For example:

```
LSF_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Can also be defined from the command line.

Valid values

Valid log classes are:

- LC_AFS and LC2_AFS: Log AFS messages
- LC_AUTH and LC2_AUTH: Log authentication messages
- LC_CHKPNT and LC2_CHKPNT: Log checkpointing messages
- LC_COMM and LC2_COMM: Log communication messages
- LC_DCE and LC2_DCE: Log messages pertaining to DCE support
- LC_EEVENTD and LC2_EEVENTD: Log eeventd messages
- LC_ELIM and LC2_ELIM: Log ELIM messages
- LC_EXEC and LC2_EXEC: Log significant steps for job execution
- LC_FAIR - Log fairshare policy messages
- LC_FILE and LC2_FILE: Log file transfer messages

- LC_HANG and LC2_HANG: Mark where a program might hang
- LC_JARRAY and LC2_JARRAY: Log job array messages
- LC_JLIMIT and LC2_JLIMIT: Log job slot limit messages
- LC_LICENSE and LC2_LICENSE : Log license management messages (LC_LICENSE is also supported for backward compatibility)
- LC_LOADINDX and LC2_LOADINDX: Log load index messages
- LC_M_LOG and LC2_M_LOG: Log multievent logging messages
- LC_MPI and LC2_MPI: Log MPI messages
- LC_MULTI and LC2_MULTI: Log messages pertaining to MultiCluster
- LC_PEND and LC2_PEND: Log messages related to job pending reasons
- LC_PERFM and LC2_PERFM: Log performance messages
- LC_PIM and LC2_PIM: Log PIM messages
- LC_PREEMPT and LC2_PREEMPT: Log preemption policy messages
- LC_RESREQ and LC2_RESREQ: Log resource requirement messages
- LC_SIGNAL and LC2_SIGNAL: Log messages pertaining to signals
- LC_SYS and LC2_SYS: Log system call messages
- LC_TRACE and LC2_TRACE: Log significant program walk steps
- LC_XDR and LC2_XDR: Log everything transferred by XDR
- LC2_KRB: Log message related to Kerberos integration
- LC2_DC: Log message related to Dynamic Cluster
- LC2_CGROUP: Log message related to cgroup operation
- LC2_TOPOLOGY: Log message related to hardware topology
- LC2_AFFINITY: Log message related to affinity
- LC2_LSF_PE: Log message related to LSF PE integration */

Default

Not defined

See also

LSF_CMD_LOG_MASK, LSF_CMD_LOGDIR, LSF_DEBUG_LIM, LSF_DEBUG_RES, LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT, LSF_LOGDIR, LSF_LIM_DEBUG, LSF_RES_DEBUG

LSF_DEBUG_LIM

Syntax

`LSF_DEBUG_LIM=log_class`

Description

Sets the log class for debugging LIM.

Specifies the log class filtering to be applied to LIM. Only messages belonging to the specified log class are recorded.

The LSF_DEBUG_LIM sets the log class and is used in combination with EGO_LOG_MASK in `ego.conf`, which sets the log level.

For example, in `ego.conf`:

```
EGO_LOG_MASK=LOG_DEBUG
```

and in `lsf.conf`:

```
LSF_DEBUG_LIM=LC_TRACE
```

Important:

If EGO is enabled, `LSF_LOG_MASK` no longer specifies LIM logging level. Use `EGO_LOG_MASK` in `ego.conf` to control message logging for LIM. The default value for `EGO_LOG_MASK` is `LOG_WARNING`.

You need to restart the daemons after setting `LSF_DEBUG_LIM` for your changes to take effect.

If you use the command `lsadmin limdebug` to temporarily change this parameter without changing `lsf.conf`, you do not need to restart the daemons.

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSF_DEBUG_LIM="LC_TRACE LC_EXEC"
```

This parameter can also be defined from the command line.

Valid values

Valid log classes are:

- `LC_AFS` and `LC2_AFS`: Log AFS messages
- `LC_AUTH` and `LC2_AUTH`: Log authentication messages
- `LC_CHKPNT` - log checkpointing messages
- `LC_COMM` and `LC2_COMM`: Log communication messages
- `LC_DCE` and `LC2_DCE`: Log messages pertaining to DCE support
- `LC_EXEC` and `LC2_EXEC`: Log significant steps for job execution
- `LC_FILE` and `LC2_FILE`: Log file transfer messages
- `LC_HANG` and `LC2_HANG`: Mark where a program might hang
- `LC_JGRP` - Log job group messages
- `LC_LICENSE` and `LC2_LICENSE` : Log license management messages (`LC_LICENSE` is also supported for backward compatibility)
- `LC_LICSCHEM` - Log License Scheduler messages
- `LC_MEMORY` - Log memory limit messages
- `LC_MULTI` and `LC2_MULTI`: Log messages pertaining to MultiCluster
- `LC_PIM` and `LC2_PIM`: Log PIM messages
- `LC_RESOURCE` - Log resource broker messages
- `LC_SIGNAL` and `LC2_SIGNAL`: Log messages pertaining to signals
- `LC_TRACE` and `LC2_TRACE`: Log significant program walk steps
- `LC_XDR` and `LC2_XDR`: Log everything transferred by XDR
- `LC2_TOPOLOGY`: Debug the hardware topology detection during runtime

EGO parameter**EGO_DEBUG_LIM****Default**

Not defined

See also

LSF_DEBUG_RES, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_DEBUG_RES**Syntax**

```
LSF_DEBUG_RES=log_class
```

Description

Sets the log class for debugging RES.

Specifies the log class filtering to be applied to RES. Only messages belonging to the specified log class are recorded.

LSF_DEBUG_RES sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG LSF_DEBUG_RES=LC_TRACE
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSF_DEBUG_RES="LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSF_DEBUG_RES for your changes to take effect.

If you use the command **lsadmin resdebug** to temporarily change this parameter without changing `lsf.conf`, you do not need to restart the daemons.

Valid values

For a list of valid log classes see LSF_DEBUG_LIM

Default

Not defined

See also

LSF_DEBUG_LIM, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_DEFAULT_FREQUENCY**Syntax**

```
LSF_DEFAULT_FREQUENCY=[float_number][unit]
```

Description

Sets the default CPU frequency for compute nodes when nodes start and when node has finished a job that uses a different CPU frequency. Value is a positive float number with units (GHz or MHz). If no units are set, the default is GHz. If nothing is set for this parameter, the nominal CPU frequency of the host will be used.

Default

Not defined (Nominal CPU frequency is used)

LSF_DHCP_ENV

Syntax

LSF_DHCP_ENV=y

Description

If defined, enables dynamic IP addressing for all LSF client hosts in the cluster.

Dynamic IP addressing is not supported across clusters in a MultiCluster environment.

If you set LSF_DHCP_ENV, you must also specify LSF_DYNAMIC_HOST_WAIT_TIME in order for hosts to rejoin a cluster after their IP address changes.

Tip:

After defining or changing this parameter, you must run **lsadmin reconfig** and **badmin mbdrestart** to restart all LSF daemons.

EGO parameter

EGO_DHCP_ENV

Default

Not defined

See also

LSF_DYNAMIC_HOST_WAIT_TIME

LSF_DISABLE_LSRUN

Syntax

LSF_DISABLE_LSRUN=y | Y

Description

When defined, RES refuses remote connections from **lsrun** and **lsgrun** unless the user is either an LSF administrator or root. However, the local host from which **lsrun** is executed will be exempted from this limitation as long as the local host is also the target host for **lsrun/lsgrun**. For remote execution by root, **LSF_ROOT_REX** must be defined.

For remote execution by root, temporarily define LSF_ROOT_USER=Y before performing the remote execution. When you are done, you must disable this parameter to ensure that your cluster remains secure.

Note: Defining LSF_ROOT_REX allows root to run **lsgrun**, even if LSF_DISABLE_LSRUN=y is defined.

Other remote execution commands, such as **ch** and **lsmake** are not affected.

Default

Set to Y at time of installation. Otherwise, not defined.

See also

LSF_ROOT_REX

LSF_DISCARD_LOG

Syntax

LSF_DISCARD_LOG=Y | N

Description

Specifies the behavior of the **mbatchd** and **mbschd** logging threads if the logging queue is full (that is, when the number of entries in the logging queue reaches the **LSF_LOG_QUEUE_SIZE** value). The logging queue contains the messages to be written to the log files.

If set to Y, the logging thread discards all new messages at a level lower than **LOG_WARNING** when the logging queue is full. LSF logs a summary of the discarded messages later.

If set to N, LSF automatically extends the size of the logging queue if the logging queue is full.

This parameter is ignored if **LSF_LOG_THREAD** is set to N.

Default

Y.

See also

LSF_LOG_THREAD to enable logging threads.

LSF_LOG_QUEUE_SIZE to configure the maximum number of entries in the logging queue that the logging thread uses before the logging queue is full.

LSF_DISPATCHER_LOGDIR

Syntax

LSF_DISPATCHER_LOGDIR=*path*

Description

Specifies the path to the log files for slot allocation decisions for queue-based fairshare.

If defined, LSF writes the results of its queue-based fairshare slot calculation to the specified directory. Each line in the file consists of a timestamp for the slot allocation and the number of slots allocated to each queue under its control. LSF logs in this file every minute. The format of this file is suitable for plotting with **gnuplot**.

Example

```
# clients managed by LSF

# Roma # Verona # Genova # Pisa # Venezia # Bologna

15/3      19:4:50    0 0 0 0 0 0
15/3      19:5:51    8 5 2 5 2 0
15/3      19:6:51    8 5 2 5 5 1
15/3      19:7:53    8 5 2 5 5 5
15/3      19:8:54    8 5 2 5 5 0
```

```
15/3      19:9:55    8 5 0 5 4 2
```

The queue names are in the header line of the file. The columns correspond to the allocations per each queue.

Default

Not defined

LSF_DJOB_TASK_REG_WAIT_TIME

Syntax

LSF_DJOB_TASK_REG_WAIT_TIME=*time_seconds*

Description

Allows users/admin to define a fixed timeout value to override the internal timeout set by LSF in order to avoid task registration timeout for a large parallel job.

Can be configured in `lsf.conf` or set as an environment variable of `bsub`. The environment variable will overwrite the `lsf.conf` configuration. If neither is present, LSF will use the default value. When it is specified by the environment variable or configured in `lsf.conf`, the value will be directly used by LSF without any adjusting.

Default

300 seconds.

LSF_DNS_CACHE

Syntax

LSF_DNS_CACHE=Y|y|N|n

Description

When set to Y, LSF generates a cluster-wide DNS host cache to improve cluster startup performance

To mitigate the burden on the DNS server when starting the LSF cluster, a cluster-based cache file (`$LSF_ENVDIR/.hosts.dnscache`) is used by all daemons on each host to reduce the number of times that LSF daemons directly call the DNS server.

The format of the cache file is the same as `$LSF_ENVDIR/hosts`:

```
IPAddress OfficialName AliasName
```

For shared installations, the master LIM creates the DNS host cache file (`$LSF_ENVDIR/.hosts.dnscache`) if the file does not exist. The **mbatchd** daemon periodically flushes the local host cache information into the DNS host cache file.

For non-shared installations, LIM creates the DNS host cache file (`$LSF_TMPDIR/.hosts.dnscache`) if the file does not exist. LIM periodically flushes local host cache information into the DNS host cache file.

When the IP address or hostname is changed on the DNS server side, LSF daemons can directly call the DNS server to obtain the updated information before the DNS host cache file is flushed with the updated information.

Default

N

LSF_DUALSTACK_PREFER_IPV6

Define this parameter when you want to ensure that clients and servers on dual-stack hosts use IPv6 addresses only. Setting this parameter configures LSF to sort the dynamically created address lookup list in order of AF_INET6 (IPv6) elements first, followed by AF_INET (IPv4) elements, and then others.

Syntax

LSF_DUALSTACK_PREFER_IPV6=Y | y

Description

Restriction:

IPv4-only and IPv6-only hosts cannot belong to the same cluster. In a IBM Spectrum LSF multicluster capability environment, you cannot mix IPv4-only and IPv6-only clusters.

Follow these guidelines for using IPv6 addresses within your cluster:

- Define this parameter only if your cluster
 - Includes only dual-stack hosts, or a mix of dual-stack and IPv6-only hosts, *and*
 - Does not include IPv4-only hosts or IPv4 servers running on dual-stack hosts (servers prior to LSF version 7)

Important:

Do not define this parameter for any other cluster configuration.

- Within a MultiCluster environment, do not define this parameter if any cluster contains IPv4-only hosts or IPv4 servers (prior to LSF version 7) running on dual-stack hosts.
- Applications must be engineered to work with the cluster IP configuration.
- If you use IPv6 addresses within your cluster, ensure that you have configured the dual-stack hosts correctly. For information about configuring dual-stack hosts, see *Administering IBM Spectrum LSF*.
- Define the parameter **LSF_ENABLE_SUPPORT_IPV6** in the `lsf.conf` file.

Default

Not defined. LSF sorts the dynamically created address lookup list in order of AF_INET (IPv4) elements first, followed by AF_INET6 (IPv6) elements, and then others. Clients and servers on dual-stack hosts use the first address lookup structure in the list (IPv4).

See also

LSF_ENABLE_SUPPORT_IPV6

LSF_DYNAMIC_HOST_TIMEOUT

Syntax

LSF_DYNAMIC_HOST_TIMEOUT=*time_hours*

LSF_DYNAMIC_HOST_TIMEOUT=*time_minutes*m | M

Description

Enables automatic removal of dynamic hosts from the cluster and specifies the timeout value (minimum 10 minutes). To improve performance in very large clusters, you should disable this feature and remove unwanted hosts from the `hostcache` file manually.

Specifies the length of time a dynamic host is unavailable before the master host removes it from the cluster. Each time LSF removes a dynamic host, mbatchd automatically reconfigures itself.

Valid value

The timeout value must be greater than or equal to 10 minutes.

Values below 10 minutes are set to the minimum allowed value 10 minutes; values above 100 hours are set to the maximum allowed value 100 hours.

Example

```
LSF_DYNAMIC_HOST_TIMEOUT=60
```

A dynamic host is removed from the cluster when it is unavailable for 60 hours.

```
LSF_DYNAMIC_HOST_TIMEOUT=60m
```

A dynamic host is removed from the cluster when it is unavailable for 60 minutes.

EGO parameter

EGO_DYNAMIC_HOST_TIMEOUT

Default

Not defined. Unavailable hosts are never removed from the cluster.

LSF_DYNAMIC_HOST_WAIT_TIME**Syntax**

```
LSF_DYNAMIC_HOST_WAIT_TIME=time_seconds
```

Description

Defines the length of time in seconds that a dynamic host waits communicating with the master LIM to either add the host to the cluster or to shut down any running daemons if the host is not added successfully.

Note:

To enable dynamically added hosts, the following parameters must be defined:

- LSF_DYNAMIC_HOST_WAIT_TIME in `lsf.conf`
- LSF_HOST_ADDR_RANGE in `lsf.cluster.cluster_name`

Recommended value

An integer greater than zero, up to 60 seconds for every 1000 hosts in the cluster, for a maximum of 15 minutes. Selecting a smaller value results in a quicker response time for hosts at the expense of an increased load on the master LIM.

Example

```
LSF_DYNAMIC_HOST_WAIT_TIME=60
```

A host waits 60 seconds from startup to send a request for the master LIM to add it to the cluster or to shut down any daemons if it is not added to the cluster.

Default

Not defined. Dynamic hosts cannot join the cluster.

LSF_EGO_DAEMON_CONTROL**Syntax**

```
LSF_EGO_DAEMON_CONTROL="Y" | "N"
```

Description

Enables EGO Service Controller to control LSF **res** and **sbatchd** startup. Set the value to "Y" if you want EGO Service Controller to start **res** and **sbatchd**, and restart them if they fail.

To configure this parameter at installation, set EGO_DAEMON_CONTROL in `install.config` so that **res** and **sbatchd** start automatically as EGO services.

If LSF_ENABLE_EGO="N", this parameter is ignored and EGO Service Controller is not started.

If you manually set EGO_DAEMON_CONTROL=Y after installation, you *must* configure LSF **res** and **sbatchd** startup to AUTOMATIC in the EGO configuration files `res.xml` and `sbatchd.xml` under `EGO_ESRVDIR/esc/conf/services`.

To avoid conflicts with existing LSF startup scripts, do not set this parameter to "Y" if you use a script (for example in `/etc/rc` or `/etc/inittab`) to start LSF daemons. If this parameter is not defined in `install.config` file, it takes default value of "N".

Important:

After installation, LSF_EGO_DAEMON_CONTROL alone *does not* change the start type for the **sbatchd** and **res** EGO services to AUTOMATIC in `res.xml` and `sbatchd.xml` under `EGO_ESRVDIR/esc/conf/services`. You must edit these files and set the `<sc:StartType>` parameter to AUTOMATIC.

Example

```
LSF_EGO_DAEMON_CONTROL="N"
```

Default

N (**res** and **sbatchd** are started manually or through operating system rc facility)

LSF_EGO_ENVDIR**Syntax**

```
LSF_EGO_ENVDIR=directory
```

Description

Directory where all EGO configuration files are installed. These files are shared throughout the system and should be readable from any host.

If LSF_ENABLE_EGO="N", this parameter is ignored and `ego.conf` is not loaded.

Default

`LSF_CONFDIR/ego/cluster_name/kernel`. If not defined, or commented out, `/etc` is assumed.

LSF_ELIM_DIR

Syntax

LSF_ELIM_DIR=*directory*

Description

Windows only. Directory in which the LSF **elim** binary files are installed.

Specify a file path and **elim** uses this file path to find and use the following **elim** files: **elim.bat**, **elim.cmd**, **elim.vbs**, and **elim.sh**.

This parameter allows you to specify a separate location for **elim** files that is different from the **LSF_SERVERDIR** parameter, such as a shared file system.

Default

LSF_SERVERDIR

See also

LSF_SERVERDIR, **EGO_ELIM_DIR**

LSF_ENABLE_BEAT_SERVICE

Syntax

LSF_ENABLE_BEAT_SERVICE="services ..."

Description

Enables the LSF Beats services (lsfbeat tool).

To use this feature, IBM Spectrum LSF Explorer must be deployed in your LSF cluster.

The **lsfbeat** tool integrates energy accounting into LSF. IBM Spectrum LSF Explorer uses Elasticsearch in the collection of the energy data of each host using Beats. LSF **bjobs** and **bhosts** can then query the data from IBM Spectrum LSF Explorer and display the job level or host level energy to users.

To enable the **lsfbeat** energy service, configure **LSF_ENABLE_BEAT_SERVICE="energy"** in **lsf.conf**, then run the **lsadmin limrestart all** command to start the **lsfbeat** service. To query energy data with the **bhosts** and **bjobs** commands, configure **LSF_QUERY_ES_FUNCTIONS="energy"** and **LSF_QUERY_ES_SERVERS="ip:port"** in **lsf.conf**.

Valid values

The following extension names are supported:

energy

If the **energy** function is defined, **lim** will start the **lsfbeat** daemon to collect host energy data from IBM Spectrum LSF Explorer Elasticsearch.

all

If the **all** keyword is defined, all supported **lsfbeat** services will be enabled.

Default

Not defined

Example

The following example shows the output of **bjobs -l** with **LSF_ENABLE_BEAT_SERVICE="energy"**:

```

user1@ibmgpu01-593: bjobs -l

Job <1933>, User <user1>, Project <default>, Status <RUN>, Queue <ex>, Command <
sleep 100000>
Thu Nov 16 03:55:14: Submitted from host <ibmgpu01>, CWD </testing/qa4/user1/egy
_test/conf>, Exclusive Execution;
Thu Nov 16 03:55:15: Started on <ibmgpu01>, Execution Home </home/user1>, Execut
ion CWD </testing/qa4/user1/egy_test/conf>;
Thu Nov 16 04:07:59: Resource usage collected.
MEM: 2 Mbytes; SWAP: 0 Mbytes; NTHREAD: 4
PGID: 3611; PIDs: 3611 3612 3615

MEMORY USAGE:
MAX MEM: 2 Mbytes; AVG MEM: 2 Mbytes

ENERGY USAGE:
163209.00 Joule/ 0.0453 kWh

```

The following example shows the output of `bjobs -o` with `LSF_ENABLE_BEAT_SERVICE="energy"`:

```

[user1@ibmgpu01 conf]$ bjobs -o energy 1933
ENERGY
163209.00 Joule

```

The following example shows the output of `bhosts -l` with `LSF_ENABLE_BEAT_SERVICE="energy"`:

```

user2@power1976>$ bhosts -l
HOST power1976
STATUS CPUF JL/U MAX NJOBS RUN SSUSP USUSP RSV DISPATCH_WINDOW
closed_Exc1 250.00 - 20 1 1 0 0 0 -

CURRENT LOAD USED FOR SCHEDULING:
          r15s  r1m  r15m  ut  pg  io  ls  it  tmp  swp  mem  slots
Total    0.0  0.0  0.1  1%  0.0  27  3  0  4173M  3.8G  105.9G  19
Reserved 0.0  0.0  0.0  0%  0.0  0  0  0  0M  0M  0M  -

LOAD THRESHOLD USED FOR SCHEDULING:
          r15s  r1m  r15m  ut  pg  io  ls  it  tmp  swp  mem
loadSched -  -  -  -  -  -  -  -  -  -  -
loadStop  -  -  -  -  -  -  -  -  -  -  -

CONFIGURED AFFINITY CPU LIST: all

ENERGY:
Current Power: 580.00 Watt Energy Consumed: 4405725.00 Joule/ 1.2238 kWh

```

See also

LSF_QUERY_ES_FUNCTIONS

LSF_ENABLE_EGO

Syntax

```
LSF_ENABLE_EGO="Y" | "N"
```

Description

Enables EGO functionality in the LSF cluster.

If you set `LSF_ENABLE_EGO="Y"`, you must set or uncomment `LSF_EGO_ENVDIR` in `lsf.conf`.

If you set `LSF_ENABLE_EGO="N"` you must remove or comment out `LSF_EGO_ENVDIR` in `lsf.conf`.

Set the value to "N" if you do not want to take advantage of the following LSF features that depend on EGO:

- LSF daemon control by EGO Service Controller
- EGO-enabled SLA scheduling

Important:

After changing the value of LSF_ENABLE_EGO, you must shut down and restart the cluster.

Default

Y (EGO is enabled in the LSF cluster)

LSF_ENABLE_EXTSCHEULER

Syntax

LSF_ENABLE_EXTSCHEULER=y | Y

Description

If set, enables mbatchd external scheduling for LSF HPC features.

Default

Set to Y at time of installation for the PARALLEL configuration template. Otherwise, not defined.

LSF_ENABLE_SUPPORT_IPV6

Syntax

LSF_ENABLE_SUPPORT_IPV6=y | Y

Description

If set, enables the use of IPv6 addresses in addition to IPv4.

Default

Not defined

See also

LSF_DUALSTACK_PREFER_IPV6

LSF_ENABLE_TMP_UNIT

Syntax

LSF_ENABLE_TMP_UNIT=Y | y | N | n

Description

The default value is LSF_ENABLE_TMP_UNIT=N | n. By default, the **LSF_UNIT_FOR_LIMITS** parameter does not affect the tmp resource, and the unit of the tmp resource for limits is MB.

When the LSF_ENABLE_TMP_UNIT=Y parameter is specified, the units that are defined by the **LSF_UNIT_FOR_LIMITS** parameter also apply cluster-wide to the tmp resource. Applying limit units to the tmp resource affects the following limits and resource requirements:

- Resource requirements at the job-level (**bsub**), queue-level (lsb.queues), and application-level (lsb.applications).
- The **TMP** resource in the Limit section and **RES_SELECT** in the GuaranteedResourcePool section and the HostExport section in the lsb.resources file.
- The load_index threshold in the lsb.queues and lsb.hosts file.

Example

When the **LSF_ENABLE_TMP_UNIT=Y** is set, and the unit for limits is specified as GB (**LSF_UNIT_FOR_LIMIT=GB**), the following job submission means mem=1 GB, swp=2 GB, tmp=3 GB, and tmp>3 GB":

```
bsub -R "rusage[mem=1,swp=2,tmp=3] select[tmp>3]"
```

Important:

Before you enable limits for the tmp resource, completely drain the cluster of all workload so that no running, pending, or finished jobs are in the system.

In the LSF multicluster capability environment, configure the same unit for all clusters.

Default

N|n

LSF_ENV_OVERRIDE

Specifies whether environment variables can override the parameter settings in the `lsf.conf` file.

Syntax

LSF_ENV_OVERRIDE=Y|y|N|n

Description

UNIX only.

Disabling this parameter prevents users' environment variables from overriding the parameter settings in the `lsf.conf` file, and LSF uses a fixed location for the `lsf.conf` file (`/etc/lsf.conf`).

If **LSF_ENV_OVERRIDE** is set to Y|y, environment variables can override values in `lsf.conf` and LSF reads the `$LSF_ENVDIR/lsf.conf` file for parameter settings.

If **LSF_ENV_OVERRIDE** is set to N|n, LSF uses only the `/etc/lsf.conf` file for parameter settings. The `$LSF_ENVDIR/lsf.conf` file is ignored. The **LSF_SERVERDIR**, **LSF_BINDIR**, and **LSF_LIBDIR** parameters must be defined to use this setting.

Default

Y.

Related reference

LSF_BINDIR

LSF_LIBDIR

Related information

LSF_SERVERDIR

LSF_ENVDIR**Syntax**

LSF_ENVDIR=*directory*

Description

Directory containing the `lsf.conf` file.

lsf.conf

By default, `lsf.conf` is installed by creating a shared copy in `LSF_CONFDIR` and adding a symbolic link from `/etc/lsf.conf` to the shared copy. If `LSF_ENVDIR` is set, the symbolic link is installed in `LSF_ENVDIR/lsf.conf`.

The `lsf.conf` file is a global environment configuration file for all LSF services and applications. The LSF default installation places the file in `LSF_CONFDIR`.

Default

`/etc`

LSF_EVENT_PROGRAM

Syntax

`LSF_EVENT_PROGRAM=event_program_name`

Description

Specifies the name of the LSF event program to use.

If a full path name is not provided, the default location of this program is `LSF_SERVERDIR`.

If a program that does not exist is specified, event generation does not work.

If this parameter is not defined, the default name is `genevent` on UNIX, and `genevent.exe` on Windows.

Default

Not defined

LSF_EVENT_RECEIVER

Syntax

`LSF_EVENT_RECEIVER=event_receiver_program_name`

Description

Specifies the LSF event receiver and enables event generation.

Any string may be used as the LSF event receiver; this information is not used by LSF to enable the feature but is only passed as an argument to the event program.

If `LSF_EVENT_PROGRAM` specifies a program that does not exist, event generation does not work.

Default

Not defined. Event generation is disabled

LSF_EXT_SERVERDIR

Directory in which the **eauth** and **esub** binary files are located.

Syntax

`LSF_EXT_SERVERDIR=directory`

Description

Specify a secure file path and LSF uses this file path to find and use the `eauth` and `esub.application` binary files. If this parameter is configured, LSF clients and daemons run `eauth` and `esub.application`

binary files in this directory only, so all `eauth` and `esub.application` binary files must be in this location. This ensures that the binary files are always run from a secure, authorized location.

Default

LSF_SERVERDIR parameter.

LSF_GET_CONF

Syntax

LSF_GET_CONF=lim

Description

Synchronizes a local host's cluster configuration with the master host's cluster configuration. Specifies that a slave host must request cluster configuration details from the LIM of a host on the SERVER_HOST list. Use when a slave host does not share a filesystem with master hosts, and therefore cannot access cluster configuration.

Default

Not defined.

LSF_GPU_AUTOCONFIG

Syntax

LSF_GPU_AUTOCONFIG=Y|N

Description

LSF_GPU_AUTOCONFIG controls whether LSF enables use of GPU resources automatically.

If set to Y, LSF automatically configures built-in GPU resources and automatically detects GPUs.

If set to N, manual configuration of GPU resources is required to use GPU features in LSF.

Whether **LSF_GPU_AUTOCONFIG** is set to Y or N, LSF will always collect GPU metrics from hosts.

The **lsload -l** command does **not** show GPU metrics. GPU metrics can be viewed using the **lsload -gpu** command, **lsload -gpuload** command, and **lshosts -gpu** command.

Note: On LSF systems using RHEL, version 7 or higher is required to support **LSF_GPU_AUTOCONFIG**.

Default

N

LSF_HOST_CACHE_DISABLE

Syntax

LSF_HOST_CACHE_DISABLE=Y|y|N|n

Description

LSF_HOST_CACHE_DISABLE=Y disables the use of a cluster-wide DNS host cache file (`$LSF_ENVDIR/.hosts.dnscache`) that can be used by all daemons on each host to reduce the number of times that LSF daemons directly call the DNS server as well as the DNS cache inside each daemon.

Default

N

LSF_HOST_CACHE_NTTL

Negative-time-to-live value in seconds.

Syntax

LSF_HOST_CACHE_NTTL=*time_seconds*

Description

Specifies the length of time the system caches a failed DNS lookup result. If you set this value to 0, LSF does not cache the result.

Note: Setting this parameter does not affect the positive-time-to-live value set by the parameter LSF_HOST_CACHE_PTTL.

Valid values

Positive integer. Set a value between 0 and 60.

Default

20 seconds

60 seconds

See also

LSF_HOST_CACHE_PTTL

LSF_HOST_CACHE_PTTL

Syntax

LSF_HOST_CACHE_PTTL=*time_seconds*

Description

Positive-time-to-live value in seconds. Specifies the length of time the system caches a successful DNS lookup result. If you set this value to zero (0), LSF does not cache the result.

Note:

Setting this parameter does not affect the negative-time-to-live value set by the parameter LSF_HOST_CACHE_NTTL.

Valid values

Positive integer. Recommended value equal to or greater than 3600 seconds (1 hour).

Default

86400 seconds (24 hours)

See also

LSF_HOST_CACHE_NTTL

LSF_HOST_MAX_GPU

Syntax

LSF_HOST_MAX_GPU=*integer*

Description

Sets the maximum number of GPU resources for all hosts in the LSF cluster.

Valid values

Integers between 1 and 32, inclusive.

Default

8

LSF_HPC_EXTENSIONS

Syntax

LSF_HPC_EXTENSIONS="*extension_name ...*"

Description

Enables LSF HPC extensions.

After adding or changing LSF_HPC_EXTENSIONS, use **badmin mbdrestart** and **badmin hrestart** to reconfigure your cluster.

Valid values

The following extension names are supported:

CUMULATIVE_RUSAGE: When a parallel job script runs multiple commands, resource usage is collected for jobs in the job script, rather than being overwritten when each command is executed.

DISP_RES_USAGE_LIMITS: **bjobs** displays resource usage limits configured in the queue as well as job-level limits.

HOST_RUSAGE: For parallel jobs, reports the correct rusage based on each host's usage and the total rusage being charged to the execution host. This host rusage breakdown applies to the blaunch framework, the pam framework, and vendor MPI jobs. For a running job, you will see run time, memory, swap, utime, stime, and pids and pgids on all hosts that a parallel job spans. For finished jobs, you will see memory, swap, utime, and stime on all hosts that a parallel job spans. The host-based rusage is reported in the JOB_FINISH record of `lsb.acct` and `lsb.stream`, and the JOB_STATUS record of `lsb.events` if the job status is done or exit. Also for finished jobs, `bjobs -l` shows CPU time, `bhist -l` shows CPU time, and `bacct -l` shows utime, stime, memory, and swap. In the MultiCluster lease model, the parallel job must run on hosts that are all in the same cluster. If you use the jobFinishLog API, all external tools must use jobFinishLog built with LSF 9.1 or later, or host-based rusage will not work. If you add or remove this extension, you must restart **mbatchd**, **sbatchd**, and **res** on all hosts. The behaviour used to be controlled by HOST_RUSAGE prior to LSF 9.1.

NO_CUMULATIVE_RUSAGE: Turn on this parameter if you do not want to collect the total resource usage for jobs that are run in a parallel job script that runs multiple commands. Instead, the resource usage is overwritten when each command is executed.

NO_HOST_RUSAGE: Turn on this parameter if you do not want to see host-based job resource usage details. However, **mbatchd** will continue to report job rusage with **bjobs** for all running jobs even if you configure **NO_HOST_RUSAGE** and restart all the daemons.

LSB_HCLOSE_BY_RES: If **res** is down, host is closed with a message

```
Host is closed because RES is not available.
```

The status of the closed host is closed_Adm. No new jobs are dispatched to this host, but currently running jobs are not suspended.

RESERVE_BY_STARTTIME: LSF selects the reservation that gives the job the earliest predicted start time.

By default, if multiple host groups are available for reservation, LSF chooses the largest possible reservation based on number of slots.

SHORT_EVENTFILE: Compresses long host name lists when event records are written to `lsb.events` and `lsb.acct` for large parallel jobs. The short host string has the format:

```
number_of_hosts*real_host_name
```

Tip:

When SHORT_EVENTFILE is enabled, older daemons and commands (pre-LSF Version 7) cannot recognize the `lsb.acct` and `lsb.events` file format.

For example, if the original host list record is

```
6 "hostA" "hostA" "hostA" "hostA" "hostB" "hostC"
```

redundant host names are removed and the short host list record becomes

```
3 "4*hostA" "hostB" "hostC"
```

When LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is set, and LSF reads the host list from `lsb.events` or `lsb.acct`, the compressed host list is expanded into a normal host list.

SHORT_EVENTFILE affects the following events and fields:

- JOB_START in `lsb.events` when a normal job is dispatched
 - numExHosts (%d)
 - execHosts (%s)
- JOB_CHUNK in `lsb.events` when a job is inserted into a job chunk
 - numExHosts (%d)
 - execHosts (%s)
- JOB_FORWARD in `lsb.events` when a job is forwarded to a MultiCluster leased host
 - numReserHosts (%d)
 - reserHosts (%s)
- JOB_FINISH record in `lsb.acct`
 - numExHosts (%d)
 - execHosts (%s)

SHORT_PIDLIST: Shortens the output from **bjobs** to omit all but the first process ID (PID) for a job. **bjobs** displays only the first ID and a count of the process group IDs (PGIDs) and process IDs for the job.

Without SHORT_PIDLIST, **bjobs -l** displays all the PGIDs and PIDs for the job. With SHORT_PIDLIST set, **bjobs -l** displays a count of the PGIDs and PIDs.

TASK_MEMLIMIT: Enables enforcement of a memory limit (**bsub -M**, **bmod -M**, or MEMLIMIT in `lsb.queues`) for individual tasks in a parallel job. If any parallel task exceeds the memory limit, LSF terminates the entire job.

Example bjobs -l output without SHORT_PIDLIST:

bjobs -l displays all the PGIDs and PIDs for the job:

```
bjobs -l
Job <109>, User <user3>, Project <default>, Status <RUN>, Queue <normal>, Interactive mode, Command <./myjob.sh>
Mon Jul 21 20:54:44 2009: Submitted from host <hostA>, CWD <${HOME}/LSF/jobs;

RUNLIMIT
10.0 min of hostA

STACKLIMIT CORELIMIT MEMLIMIT
5256 K 10000 K 5000 K
Mon Jul 21 20:54:51 2009: Started on <hostA>;
Mon Jul 21 20:55:03 2009: Resource usage collected.
MEM: 2 Mbytes; SWAP: 15 Mbytes
PGID: 256871; PIDs: 256871
PGID: 257325; PIDs: 257325 257500 257482 257501 257523
257525 257531

SCHEDULING PARAMETERS:
      r15s  r1m  r15m  ut      pg    io    ls    it    tmp    swp    mem
loadSched -    -    -    -    -    -    -    -    -    -    -
loadStop  -    -    -    -    -    -    -    -    -    -    -
      cpuspeed  bandwidth
loadSched -    -
loadStop  -    -
<< Job <109> is done successfully. >>
```

Example bjobs -l output with SHORT_PIDLIST:

bjobs -l displays a count of the PGIDS and PIDs:

```
bjobs -l
Job <109>, User <user3>, Project <default>, Status <RUN>, Queue <normal>, Interactive mode, Command <./myjob.sh>
Mon Jul 21 20:54:44 2009: Submitted from host <hostA>, CWD <${HOME}/LSF/jobs;

RUNLIMIT
10.0 min of hostA

STACKLIMIT CORELIMIT MEMLIMIT
5256 K 10000 K 5000 K
Mon Jul 21 20:54:51 2009: Started on <hostA>;
Mon Jul 21 20:55:03 2009: Resource usage collected.
MEM: 2 Mbytes; SWAP: 15 Mbytes
PGID(s): 256871:1 PID, 257325:7 PIDs

SCHEDULING PARAMETERS:
      r15s  r1m  r15m  ut      pg    io    ls    it    tmp    swp    mem
loadSched -    -    -    -    -    -    -    -    -    -    -
loadStop  -    -    -    -    -    -    -    -    -    -    -
      cpuspeed  bandwidth
loadSched -    -
loadStop  -    -
```

Default

Set to "CUMULATIVE_RUSAGE HOST_RUSAGE LSB_HCLOSE_BY_RES SHORT_EVENTFILE" at time of installation for the PARALLEL configuration template. If otherwise undefined, then "HOST_RUSAGE CUMULATIVE_RUSAGE".

LSF_HPC_PJL_LOADENV_TIMEOUT**Syntax**

LSF_HPC_PJL_LOADENV_TIMEOUT=*time_seconds*

Description

Timeout value in seconds for PJL to load or unload the environment. For example, set LSF_HPC_PJL_LOADENV_TIMEOUT to the number of seconds needed for IBM POE to load or unload adapter windows.

At job startup, the PJL times out if the first task fails to register with PAM within the specified timeout value. At job shutdown, the PJL times out if it fails to exit after the last Taskstarter termination report within the specified timeout value.

Default

LSF_HPC_PJL_LOADENV_TIMEOUT=300

LSF_HWLOC_DYNAMIC

Enables LSF to dynamically load the hardware locality (**hwloc**) library from system library paths to detect newer hardware.

Syntax

LSF_HWLOC_DYNAMIC=y|Y|n|N

Description

This parameter allows you to use the latest version of the **hwloc** integration at any time if there are no compatibility issues between the latest versions of the **hwloc** library and header file for **hwloc**, Version 1.11.8. If LSF fails to load the library, LSF defaults to using the **hwloc** functions in the static library.

Note: When using an **hwloc** header file with a different version of the **hwloc** library, there might be compatibility issues that cause LSF binary files (such as LIM) to crash. When enabling the dynamic load method and the library version in the system library path is older than the header that LSF used, LSF defaults to using functions in the static library.

Default

N

LSF_ID_PORT**Syntax**

LSF_ID_PORT=*port_number*

Description

The network port number used to communicate with the authentication daemon when LSF_AUTH is set to ident.

Default

Not defined

LSF_IMAGE_INFO_EXPIRE_INTERVAL

Specifies the amount of time, in seconds, that the host image information is valid in **mosquitto**.

Syntax

LSF_IMAGE_INFO_EXPIRE_INTERVAL=*seconds*

Description

If the container image information is older than this specified value, the image information expires.

Default

86400 seconds (1 day)

LSF_IMAGE_INFO_PUBLISH_INTERVAL

Specifies the interval, in seconds, for the **lim** process to fork a new process to collect host Docker container image information.

Syntax

LSF_IMAGE_INFO_PUBLISH_INTERVAL=*seconds*

Valid values

Specify an integer between 10 and 600.

Default

60 seconds

LSF_INCLUDEDIR**Syntax**

LSF_INCLUDEDIR=*directory*

Description

Directory under which the LSF API header files `lsf.h` and `lsbatch.h` are installed.

Default

LSF_INDEP/include

See also

LSF_INDEP

LSF_INDEP**Syntax**

LSF_INDEP=*directory*

Description

Specifies the default top-level directory for all machine-independent LSF files.

This includes man pages, configuration files, working directories, and examples. For example, defining `LSF_INDEP` as `/usr/share/lsf/mnt` places man pages in `/usr/share/lsf/mnt/man`, configuration files in `/usr/share/lsf/mnt/conf`, and so on.

The files in `LSF_INDEP` can be shared by all machines in the cluster.

As shown in the following list, `LSF_INDEP` is incorporated into other LSF environment variables.

- `LSB_SHAREDIR=$LSF_INDEP/work`

- LSF_CONFDIR=\$LSF_INDEP/conf
- LSF_INCLUDEDIR=\$LSF_INDEP/include
- LSF_MANDIR=\$LSF_INDEP/man
- XLSF_APPDIR=\$LSF_INDEP/misc

Default

/usr/share/lsf/mnt

See also

LSF_MACHDEP, LSB_SHAREDIR, LSF_CONFDIR, LSF_INCLUDEDIR, LSF_MANDIR, XLSF_APPDIR

LSF_INTELLIGENT_CPU_BIND

Enables LSF to bind a defined set of LSF daemons to CPUs.

Syntax

LSF_INTELLIGENT_CPU_BIND=Y|y|N|n

Description

If this parameter is set to Y or y, LSF automatically detects the topology of each master host and master candidate, binds the LSF daemons according to the automatic binding policy, and generates a CPU binding configuration file for that host. These configuration files are in the *LSF_ENVDIR/cpu_binding* directory and the file names are in the format of *host_name.cluster_name* and are in the *LSF_ENVDIR/cpu_binding* directory. LSF binds LSF daemons to CPU cores according to this CPU binding configuration file.

If the CPU binding configuration file no longer exists for the host, LIM automatically detects the topology of that host again and binds the LSF daemons according to the automatic binding policy.

The CPU binding configuration file specifies the CPU binding for the following LSF daemons:

- **MBATCHD_MAIN**: The main **mbatchd** daemon thread.
- **MBATCHD_ESTRV**: The **mbatchd** daemon thread for the simulation-based job start time estimator, if enabled.
- **MBATCHD_CONN**: The **mbatchd** daemon thread for the long connection process.
- **MBATCHD_LOG**: The thread that is shared between the switch log and the **elogd** daemon.
- **LIM_MAIN**: The main LIM thread.
- **LIM_UDP_THREAD**: LIM requests that use the UDP protocol.
- **MBSCHD_MAIN**: The main **mbschd** daemon thread.
- **MBSCHD_ESTIMATOR**: The **mbschd** daemon thread for the simulation-based job start time estimator, if enabled.

For the topology and CPU list on the host, run the **lim -T** command.

If the CPU binding configuration file is changed, you must restart the LSF daemons by running the following commands in the following order to apply the new CPU binding settings:

1. **lsadmin limrestart**
2. **badm hrestart**
3. **badm mbdrestart**

Note: CPU binding configuration files do not support comments. LSF removes any comments in the CPU binding configuration files after LIM restarts.

The **LSF_INTELLIGENT_CPU_BIND** parameter takes precedence over other CPU binding parameters. If you enable **LSF_INTELLIGENT_CPU_BIND**, LSF ignores the following parameters:

- **EGO_DAEMONS_CPUS**
- **ESTIMATOR_CPUS**
- **LSF_DAEMONS_CPUS**
- **MBD_QUERY_CPUS**

Note: This parameter is ignored if the master and master candidate hosts have fewer than eight CPU cores.

Default

N.

LSF_INTERACTIVE_STDERR**Syntax**

LSF_INTERACTIVE_STDERR=y | n

Description

Separates `stderr` from `stdout` for interactive tasks and interactive batch jobs.

This is useful to redirect output to a file with regular operators instead of the **bsub -e *err_file*** and **-o *out_file*** options.

This parameter can also be enabled or disabled as an environment variable.

**CAUTION:**

If you enable this parameter globally in `lsf.conf`, check any custom scripts that manipulate `stderr` and `stdout`.

When this parameter is not defined or set to `n`, the following are written to `stdout` on the submission host for interactive tasks and interactive batch jobs:

- Job standard output messages
- Job standard error messages

The following are written to `stderr` on the submission host for interactive tasks and interactive batch jobs:

- LSF messages
- NIOS standard messages
- NIOS debug messages (if `LSF_NIOS_DEBUG=1` in `lsf.conf`)

When this parameter is set to `y`, the following are written to `stdout` on the submission host for interactive tasks and interactive batch jobs:

- Job standard output messages

The following are written to `stderr` on the submission host:

- Job standard error messages
- LSF messages
- NIOS standard messages
- NIOS debug messages (if `LSF_NIOS_DEBUG=1` in `lsf.conf`)

Default

Not defined

Notes

When this parameter is set, the change affects interactive tasks and interactive batch jobs run with the following commands:

- **bsub -I**
- **bsub -Ip**
- **bsub -Is**
- **lsrun**
- **lsgrun**
- **lsmake** (makefile)
- `bsub pam` (HPC features must be enabled)

Limitations

- Pseudo-terminal: Do not use this parameter if your application depends on `stderr` as a terminal. This is because LSF must use a non-pseudo-terminal connection to separate `stderr` from `stdout`.
- Synchronization: Do not use this parameter if you depend on messages in `stderr` and `stdout` to be synchronized and jobs in your environment are continuously submitted. A continuous stream of messages causes `stderr` and `stdout` to not be synchronized. This can be emphasized with parallel jobs. This situation is similar to that of `rsh`.
- NIOS standard and debug messages: NIOS standard messages, and debug messages (when `LSF_NIOS_DEBUG=1` in `lsf.conf` or as an environment variable) are written to `stderr`. NIOS standard messages are in the format `<<message>>`, which makes it easier to remove them if you wish. To redirect NIOS debug messages to a file, define **LSF_CMD_LOGDIR** in `lsf.conf` or as an environment variable.

See also

LSF_NIOS_DEBUG, LSF_CMD_LOGDIR

LSF_LD_SECURITY**Syntax**

`LSF_LD_SECURITY=y | n`

Description

LSF_LD_SECURITY: When set, jobs submitted using **bsub -Is** or **bsub -Ip** cause the environment variables `LD_PRELOAD` and `LD_LIBRARY_PATH` to be removed from the job environment during job initialization to ensure enhanced security against users obtaining root privileges.

Two new environment variables are created (`LSF_LD_LIBRARY_PATH` and `LSF_LD_PRELOAD`) to allow `LD_PRELOAD` and `LD_LIBRARY_PATH` to be put back before the job runs.

Default

N

LSF_LIBDIR

Syntax

LSF_LIBDIR=*directory*

Description

Specifies the directory in which the LSF libraries are installed. Library files are shared by all hosts of the same type.

Default

LSF_MACHDEP/lib

LSF_LIC_SCHED_CLUSTER_ALIAS

Defines an alias for a cluster name and allows LSF License Scheduler to manage multiple clusters that share a same cluster name.

Syntax

LSF_LIC_SCHED_CLUSTER_ALIAS="*cluster_name_alias*"

Description

LSF uses the specified alias as the cluster name when it communicates with LSF License Scheduler. The `lsf.licensescheduler` configuration file and commands, such as **blstat** and **blinfo**, refer to the cluster by the specified alias.

After you change the parameter value, run the **badmin mbdrestart** command to make your changes take effect.

Note: The **LSF_LIC_SCHED_CLUSTER_ALIAS** parameter is not supported in clusters with the LSF/XL feature enabled.

Use the **badmin showconf mbd** command to see the setting of the **LSF_LIC_SCHED_CLUSTER_ALIAS** parameter.

Valid values

Valid values are strings with characters that are either alphanumeric characters, underscores (`_`), or hyphens (`-`). The length of this string cannot exceed 40 characters.

Default

Not defined

LSF_LIC_SCHED_HOSTS

Syntax

LSF_LIC_SCHED_HOSTS="*candidate_host_list*"

candidate_host_list is a space-separated list of hosts that are candidate License Scheduler hosts.

Description

The candidate License Scheduler host list is read by LIM on each host to check if the host is a candidate License Scheduler master host. If the host is on the list, LIM starts the License Scheduler daemon (bld) on the host.

LSF_LIC_SCHED_PREEMPT_REQUEUE

Set this parameter to requeue a job whose license is preempted by IBM Spectrum LSF License Scheduler. The job is killed and requeued instead of suspended.

Syntax

LSF_LIC_SCHED_PREEMPT_REQUEUE=y | n

Description

If you set the **LSF_LIC_SCHED_PREEMPT_REQUEUE** parameter, do not set the **LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE** parameter. If both parameters are set, the **LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE** parameter is ignored.

Default

N

See also

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE, **LSF_LIC_SCHED_PREEMPT_STOP**

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE

Syntax

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE=y | n

Description

Set this parameter to release the resources of a LSF License Scheduler job that is suspended. These resources are only available to pending LSF License Scheduler jobs that request at least one license that is the same as the suspended job.

If the suspended LSF License Scheduler job is an exclusive job (-x), the suspended job's resources are not released and pending LSF License Scheduler jobs cannot use these resources. A pending exclusive LSF License Scheduler job can also be blocked from starting on a host by a suspended LSF License Scheduler job, even when the jobs share licenses or exclusive preemption is enabled.

Jobs attached to a service class are not normally preemptable with queue-based preemption, except by jobs in queues with **SLA_GUARANTEES_IGNORE=Y** specified (in `lsb.queues`). However, LSF License Scheduler can preempt jobs with service classes for licenses.

By default, the job slots are the only resources available after a job is suspended. You can also specify that memory resources are available by enabling preemption for memory resources. To enable memory resource preemption, specify **PREEMPTABLE_RESOURCES = mem** in `lsb.params`.

By default, the job slots are the only resources available after a job is suspended. You can also specify that memory or affinity resources are available by enabling preemption for these resources:

- To enable memory resource preemption, specify **PREEMPTABLE_RESOURCES = mem** in `lsb.params`.
- To enable affinity resource preemption, specify **PREEMPT_JOBTYPE = AFFINITY** in `lsb.params`.

If you set **LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE**, do not set **LSF_LIC_SCHED_PREEMPT_REQUEUE**. If both these parameters are set, **LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE** is ignored.

Default

Y

See also

LSF_LIC_SCHED_PREEMPT_REQUEUE, LSF_LIC_SCHED_PREEMPT_STOP

LSF_LIC_SCHED_PREEMPT_STOP

Syntax

LSF_LIC_SCHED_PREEMPT_STOP=y | n

Description

Set this parameter to use job controls to stop a job that is preempted. When this parameter is set, a UNIX SIGSTOP signal is sent to suspend a job instead of a UNIX SIGTSTP.

To send a SIGSTOP signal instead of SIGTSTP, the following parameter in `lsb.queues` must also be set:

```
JOB_CONTROLS=SUSPEND[SIGSTOP]
```

Default

N

See also

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE, LSF_LIC_SCHED_PREEMPT_REQUEUE

LSF_LIC_SCHED_STRICT_PROJECT_NAME

Syntax

LSF_LIC_SCHED_STRICT_PROJECT_NAME=y | n

Description

Enforces strict checking of the License Scheduler project name upon job submission or job modification (**bsub** or **bmod**). If the project named is misspelled (case sensitivity applies), the job is rejected.

If this parameter is not set or it is set to n, and if there is an error in the project name, the default project is used.

Default

N

LSF_LIM_API_NTRIES

Syntax

LSF_LIM_API_NTRIES=*integer*

Description

Defines the number of times LSF commands will try to communicate with the LIM API when LIM is not available. LSF_LIM_API_NTRIES is ignored by LSF and EGO daemons and EGO commands. The LSF_LIM_API_NTRIES environment variable overrides the value of LSF_LIM_API_NTRIES in `lsf.conf`.

Valid values

1 to 65535

Default

1. LIM API exits without retrying.

LSF_LIM_DEBUG**Syntax**

LSF_LIM_DEBUG=1 | 2

Description

Sets LSF to debug mode.

If LSF_LIM_DEBUG is defined, LIM operates in single user mode. No security checking is performed, so LIM should not run as root.

LIM does not look in the services database for the LIM service port number. Instead, it uses port number 36000 unless LSF_LIM_PORT has been defined.

Specify 1 for this parameter unless you are testing LSF.

Valid values

LSF_LIM_DEBUG=1

LIM runs in the background with no associated control terminal.

LSF_LIM_DEBUG=2

LIM runs in the foreground and prints error messages to `tty`.

EGO parameter

EGO_LIM_DEBUG

Default

Not defined

See also

LSF_RES_DEBUG, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_LIM_IGNORE_CHECKSUM**Syntax**

LSF_LIM_IGNORE_CHECKSUM=y | Y

Description

Configure LSF_LIM_IGNORE_CHECKSUM=Y to ignore warning messages logged to **lim** log files on non-master hosts.

When LSF_MASTER_LIST is set, **lsadmin reconfig** only restarts master candidate hosts (for example, after adding or removing hosts from the cluster). This can cause superfluous warning messages like the following to be logged in the **lim** log files for non-master hosts because **lim** on these hosts are not restarted after configuration change:

```
Aug 26 13:47:35 2006 9746 4 10.1 xdr_loadvector:
Sender <10.225.36.46:9999> has a different configuration
```

Default

Not defined.

See also

LSF_MASTER_LIST

LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT

Syntax

LSF_LIM_PORT=*port_number*

Description

TCP service ports to use for communication with the LSF daemons.

If port parameters are not defined, LSF obtains the port numbers by looking up the LSF service names in the /etc/services file or the NIS (UNIX). If it is not possible to modify the services database, you can define these port parameters to set the port numbers.

EGO parameter

EGO_LIM_PORT

Default

On UNIX, the default is to get port numbers from the services database.

On Windows, these parameters are mandatory.

Default port number values are:

- LSF_LIM_PORT=7869
- LSF_RES_PORT=6878
- LSB_MBD_PORT=6881
- LSB_SBD_PORT=6882

LSF_LIM_STARTUP_RETRY_INTERVAL

Syntax

LSF_LIM_STARTUP_RETRY_INTERVAL=*seconds*

Description

If lim failed to initialize the socket, this parameter specifies the amount of time before lim tries again (if LSF_LIM_STARTUP_RETRY_TIMES is specified).

Valid values

1 - INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

If **LSF_LIM_STARTUP_RETRY_TIMES** is defined, the default value is 5. Otherwise, not defined.

See also**LSF_LIM_STARTUP_RETRY_TIMES****LSF_LIM_STARTUP_RETRY_TIMES****Syntax**LSF_LIM_STARTUP_RETRY_TIMES=*integer***Description**

The maximum number of times for lim to repeat attempts to initialize a socket if the first attempt failed.

Valid values

0 - INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

2

lim will re-attempt to initialize a socket up to two additional times if the first attempt failed.

See also**LSF_LIM_STARTUP_RETRY_INTERVAL****LSF_LINUX_CGROUP_ACCT****Syntax**

LSF_LINUX_CGROUP_ACCT=Y|y|N|n

Description

Enable this parameter to track processes based on CPU and memory accounting for Linux systems that support cgroup's memory and cpuacct subsystems. Once enabled, this parameter takes effect for new jobs. When this parameter and **LSF_PROCESS_TRACKING** are enabled, they take precedence over parameters **LSF_PIM_LINUX_ENHANCE** and **EGO_PIM_SWAP_REPORT**.

Default

Set to Y at time of installation. If otherwise undefined, then N.

LSF_LIVE_CONFDIR**Syntax**LSF_LIVE_CONFDIR=*directory***Description**

Enables and disables live reconfiguration (**bconf** command) and sets the directory where configuration files changed by live reconfiguration are saved. **bconf** requests will be rejected if the directory does not exist and cannot be created, or is specified using a relative path.

When **LSF_LIVE_CONFDIR** is defined and contains configuration files, all LSF restart and reconfiguration reads these configuration files instead of the files in LSF_CONFDIR.

lsf.conf

After adding or changing `LSF_LIVE_CONFDIR` in `lsf.conf`, use **badadmin mbdrestart** and **lsadmin reconfig** to reconfigure your cluster.

Important:

Remove **LSF_LIVE_CONFDIR** configuration files or merge files into **LSF_CONFDIR** before disabling **bconf**, upgrading LSF, applying patches to LSF, or adding server hosts.

See **bconf** in the *LSF Command Reference* or **bconf** man page for **bconf** (live reconfiguration) details.

Default

During installation, **LSF_LIVE_CONFDIR** is set to **LSB_SHAREDIR/cluster_name/live_confdir** where *cluster_name* is the name of the LSF cluster, as returned by **lsid**.

See also

LSF_CONFDIR, **LSB_CONFDIR**

LSF_LOAD_USER_PROFILE

Syntax

`LSF_LOAD_USER_PROFILE=local | roaming`

Description

When running jobs on Windows hosts, you can specify whether a user profile should be loaded. Use this parameter if you have jobs that need to access user-specific resources associated with a user profile.

Local and roaming user profiles are Windows features. For more information about them, check Microsoft documentation.

- Local: LSF loads the Windows user profile from the local execution machine (the host on which the job runs).

Note:

If the user has logged onto the machine before, the profile of that user is used. If not, the profile for the default user is used

- Roaming: LSF loads a roaming user profile if it has been set up. If not, the local user profile is loaded instead.

Default

Not defined. No user profiles are loaded when jobs run on Windows hosts.

LSF_LOCAL_RESOURCES

Syntax

`LSF_LOCAL_RESOURCES="resource ..."`

Description

Defines instances of local resources residing on the slave host.

- For numeric resources, defined name-value pairs:

```
"[resourcemap value*resource_name]"
```

- For Boolean resources, the value is the resource name in the form:

```
"[resource resource_name]"
```

When the slave host calls the master host to add itself, it also reports its local resources. The local resources to be added must be defined in `lsf.shared`. `Default` indicates an instance of a resource on each host in the cluster. This specifies a special case where the resource is in effect not shared and is local to every host. `Default` means at each host. Normally, you should not need to use `default`, because by default, all built-in resources are local to each host. However, resources that you defined must always be mapped. You might want to use *ResourceMap* for a non-shared static resource if you need to specify different values for the resource on different hosts.

If the same resource is already defined in `lsf.shared` as `default` or `all`, it cannot be added as a local resource. The shared resource overrides the local one.

Tip:

`LSF_LOCAL_RESOURCES` is usually set in the `slave.config` file during installation. If `LSF_LOCAL_RESOURCES` are already defined in a local `lsf.conf` on the slave host, **lsfinstall** does not add resources you define in `LSF_LOCAL_RESOURCES` in `slave.config`. You should not have duplicate `LSF_LOCAL_RESOURCES` entries in `lsf.conf`. If local resources are defined more than once, only the last definition is valid.

Important:

Resources must already be mapped to hosts in the `ResourceMap` section of `lsf.cluster.cluster_name`. If the `ResourceMap` section does not exist, local resources are not added.

Example

```
LSF_LOCAL_RESOURCES="[resourcecmap 1*verilog] [resource linux] [resource !bigmem]"
```

Prefix the resource name with an exclamation mark (!) to indicate that the resource is exclusive to the host.

EGO parameter

`EGO_LOCAL_RESOURCES`

Default

Not defined

LSF_LOG_MASK

Syntax

`LSF_LOG_MASK=message_log_level`

Description

Specifies the logging level of error messages for LSF daemons, except LIM, which is controlled by EGO.

For **mbatchd** and **mbschd**, `LSF_LOG_MASK_MBD` and `LSF_LOG_MASK_SCH` override `LSF_LOG_MASK`.

For example:

```
LSF_LOG_MASK=LOG_DEBUG
```

If EGO is enabled in the LSF cluster, and `EGO_LOG_MASK` is not defined, LSF uses the value of `LSF_LOG_MASK` for LIM, PIM, and MELIM. EGO **vemkd** and **pem** components continue to use the EGO default values. If `EGO_LOG_MASK` is defined, and EGO is enabled, then EGO value is taken.

To specify the logging level of error messages for LSF commands, use `LSF_CMD_LOG_MASK`. To specify the logging level of error messages for LSF batch commands, use `LSB_CMD_LOG_MASK`.

On UNIX, this is similar to `syslog`. All messages logged at the specified level or higher are recorded; lower level messages are discarded. The `LSF_LOG_MASK` value can be any log priority symbol that is defined in `syslog.h` (see `syslog`).

The log levels in order from highest to lowest are:

- `LOG_EMERG`
- `LOG_ALERT`
- `LOG_CRIT`
- `LOG_ERR`
- `LOG_WARNING`
- `LOG_NOTICE`
- `LOG_INFO`
- `LOG_DEBUG`
- `LOG_DEBUG1`
- `LOG_DEBUG2`
- `LOG_DEBUG3`

The most important LSF log messages are at the `LOG_ERR` or `LOG_WARNING` level. Messages at the `LOG_INFO` and `LOG_DEBUG` level are only useful for debugging.

Although message log level implements similar functionality to UNIX `syslog`, there is no dependency on UNIX `syslog`. It works even if messages are being logged to files instead of `syslog`.

LSF logs error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by `LSF_LOG_MASK` determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level `LOG_DEBUG` contains the fewest number of debugging messages and is used for basic debugging. The level `LOG_DEBUG3` records all debugging messages, and can cause log files to grow very large; it is not often used. Most debugging is done at the level `LOG_DEBUG2`.

In versions earlier than LSF 4.0, you needed to restart the daemons after setting `LSF_LOG_MASK` in order for your changes to take effect.

LSF 4.0 implements dynamic debugging, which means you do not need to restart the daemons after setting a debugging environment variable.

EGO parameter

`EGO_LOG_MASK`

Default

`LOG_WARNING`

See also

`LSB_CMD_LOG_MASK`, `LSB_CMD_LOGDIR`, `LSB_DEBUG`, `LSB_DEBUG_CMD`, `LSB_DEBUG_NQS`,
`LSB_TIME_CMD`, `LSF_CMD_LOGDIR`, `LSF_CMD_LOG_MASK`, `LSF_DEBUG_LIM`, `LSB_DEBUG_MBD`,
`LSF_DEBUG_RES`, `LSB_DEBUG_SBD`, `LSB_DEBUG_SCH`, `LSF_LOG_MASK`, `LSF_LOGDIR`, `LSF_TIME_CMD`

LSF_LOG_MASK_LIM

Syntax

`LSF_LOG_MASK_LIM=message_log_level`

Description

Specifies the logging level of error messages for LSF **LIM** only. This value overrides LSF_LOG_MASK for **LIM** only.

For example:

```
LSF_LOG_MASK_LIM=LOG_DEBUG
```

The valid log levels for this parameter are:

- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

Run **lsadmin limrestart** to make changes take effect.

Default

Not defined (logging level is controlled by LSF_LOG_MASK).

LSF_LOG_MASK_RES**Syntax**

```
LSF_LOG_MASK_RES=message_log_level
```

Description

Specifies the logging level of error messages for LSF **RES** only. This value overrides LSF_LOG_MASK for **RES** only.

For example:

```
LSF_LOG_MASK_RES=LOG_DEBUG
```

The valid log levels for this parameter are:

- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

Run **lsadmin resrestart** to make changes take effect.

Default

Not defined (logging level is controlled by LSF_LOG_MASK).

LSF_LOG_MASK_WIN

Syntax

LSF_LOG_MASK_WIN=*message_log_level*

Description

Allows you to reduce the information logged to the LSF Windows event log files. Messages of lower severity than the specified level are discarded.

For all LSF files, the types of messages saved depends on LSF_LOG_MASK, so the threshold for the Windows event logs is either LSF_LOG_MASK or LSF_LOG_MASK_WIN, whichever is higher. LSF_LOG_MASK_WIN is ignored if LSF_LOG_MASK is set to a higher level.

The LSF event log files for Windows are:

- *lim.log.host_name*
- *res.log.host_name*
- *sbatchd.log.host_name*
- *mbatchd.log.host_name*
- *pim.log.host_name*

The log levels you can specify for this parameter, in order from highest to lowest, are:

- LOG_ERR
- LOG_WARNING
- LOG_INFO
- LOG_NONE (LSF does not log Windows events)

Default

LOG_ERR

See also

LSF_LOG_MASK

LSF_LOG_QUEUE_SIZE

Specifies the maximum number of entries in the logging queue that the **mbatchd** and **mbschd** logging threads use.

Syntax

LSF_LOG_QUEUE_SIZE=*integer*

Description

The logging queue contains the messages to be written to the log files. After the number of entries in the logging queue reaches this number, the logging queue is full. When the logging queue is full, **LSF_DISCARD_LOG** specifies the behavior of the logging thread.

To temporarily change the size of the **mbatchd** logging queue, run **badmin mbddebug... -s log_queue_size**.

To temporarily change the size of the **mbschd** logging queue, run **badmin schddebug... -s log_queue_size**.

This parameter is ignored if **LSF_LOG_THREAD** is set to N.

Valid values

100 to 500000

Default

10000

See also**LSF_LOG_THREAD** to enable logging threads.**LSF_DISCARD_LOG** to specify the behavior of the logging thread if the logging queue is full.**LSF_LOG_THREAD****Syntax**

LSF_LOG_THREAD=Y | N

DescriptionEnables **mbatchd** and **mbschd** to create dedicated threads to write messages to the log files.This parameter is enabled by default to reduce the impact of logging messages on the performance of the **mbatchd** and **mbschd** daemons.**Default**

Y.

See also**LSF_LOG_QUEUE_SIZE** to configure the maximum number of entries in the logging queue that the logging thread uses before the logging queue is full.**LSF_DISCARD_LOG** to specify the behavior of the logging thread if the logging queue is full.**LSF_LOGDIR****Syntax**LSF_LOGDIR=*directory***Description**Defines the LSF system log file directory. Error messages from all servers are logged into files in this directory. To effectively use debugging, set LSF_LOGDIR to a directory such as /tmp. This can be done in your own environment from the shell or in `lsf.conf`.**Windows**

LSF_LOGDIR is required on Windows if you wish to enable logging.

You must also define LSF_LOGDIR_USE_WIN_REG=n.

If you define LSF_LOGDIR without defining LSF_LOGDIR_USE_WIN_REG=n, LSF logs error messages into files in the default local directory specified in one of the following Windows registry keys:

- On Windows 2000, Windows XP, and Windows 2003:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM Platform\LSF\LSF_LOGDIR
```

- On Windows XP x64 and Windows 2003 x64:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\IBM Platform\LSF\LSF_LOGDIR
```

If a server is unable to write in the LSF system log file directory, LSF attempts to write to the following directories in the following order:

- LSF_TMPDIR if defined
- %TMP% if defined
- %TEMP% if defined
- System directory, for example, c:\winnt

UNIX

If a server is unable to write in this directory, the error logs are created in /tmp on UNIX.

If LSF_LOGDIR is not defined, syslog is used to log everything to the system log using the LOG_DAEMON facility. The syslog facility is available by default on most UNIX systems. The /etc/syslog.conf file controls the way messages are logged and the files they are logged to. See the man pages for the syslogd daemon and the syslog function for more information.

Default

Not defined. On UNIX, log messages go to syslog. On Windows, no logging is performed.

See also

LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR_USE_WIN_REG, LSF_TIME_CMD

Files

- lim.log.*host_name*
- res.log.*host_name*
- sbatchd.log.*host_name*
- sbatchdc.log.*host_name* (when LSF_DAEMON_WRAP=Y)
- mbatchd.log.*host_name*
- eeveed.log.*host_name*
- pim.log.*host_name*

LSF_LOGDIR_USE_WIN_REG

Syntax

```
LSF_LOGDIR_USE_WIN_REG=n | N
```

Description

Windows only.

If set, LSF logs error messages into files in the directory specified by LSF_LOGDIR in lsf.conf.

Use this parameter to enable LSF to save log files in a different location from the default local directory specified in the Windows registry.

If not set, or if set to any value other than N or n, LSF logs error messages into files in the default local directory specified in one of the following Windows registry keys:

- On Windows 2000, Windows XP, and Windows 2003:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM Platform\LSF\LSF_LOGDIR
```

- On Windows XP x64 and Windows 2003 x64:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\IBM Platform\LSF\LSF_LOGDIR
```

Default

Not set.

LSF uses the default local directory specified in the Windows registry.

See also

LSF_LOGDIR

LSF_LOGFILE_OWNER

Syntax

```
LSF_LOGFILE_OWNER="user_name"
```

Description

Specifies an owner for the LSF log files other than the default, the owner of `lsf.conf`. To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME**user_name*).

Default

Not set. The LSF Administrator with root privileges is the owner of LSF log files.

LSF_LSHOSTS_FORMAT

Syntax

```
LSF_LSHOSTS_FORMAT="field_name[:[-][output_width][:unit]] ... [delimiter='character']"
```

Description

Sets the customized output format for the **lshosts** command.

- Specify which **lshosts** fields (or aliases instead of the full field names), in which order, and with what width to display.
- Specify only the **lshosts** field name or alias to set its output to unlimited width and left justification.
- Specify the colon (:) without a width to set the output width to the recommended width for that field.
- Specify the colon (:) with a width to set the maximum number of characters to display for the field. When its value exceeds this width, **lshosts** truncates the ending characters.
- Specify a hyphen (-) to set right justification when **lshosts** displays the output for the specific field. If not specified, the default is to set left justification when **lshosts** displays the output for a field.
- Specify the unit colon (:) with a unit to set the unit for the output of the specific field:
 - Specify S to use a built-in conversion for space or capacity, such as memory or disk space. Values are automatically scaled for M (MB), G (GB), and T (TB), where the default unit is M (MB).

For example, when displaying the **mem** field with a specified width of 3,

- For a value of 30, running the **lshosts -o "maxmem:3:S"** command shows 30.0M.
- For a value of 4096, running the **lshosts -o "maxswp:3:S"** command shows 4.0G.

- For a value of 5000000, running the **lshosts -o "maxtmp:3:S"** command shows 4.8T.
- Specify D to use a built-in conversion for duration or time, such as memory or disk space. Values are automatically scaled for s (seconds), m (minutes), h (hours), and d (days), where the default unit is s (seconds). The automatically scaled value is rounded up after the first decimal point.

For example, when displaying the external **mytime** resource field with a specified width of 5,

- For a value of 30, running the **lshosts -o "mytime:5:D"** command shows 30.0s.
- For a value of 8000, running the **lshosts -o "mytime:5:D"** command shows 2.2h.
- For a value of 5000000, running the **lshosts -o "mytime:5:D"** command shows 57.8d.
- Specify any other string of 1 - 3 characters and the characters are used as is in the field value. The first character must be a letter (upper or lower case). The second and third characters must be an alphanumeric character.

For example, when displaying the external **gpu_temp** resource with a width of 3, running the

lshosts -o "gpu_temp:3:C" command for a value of 30 shows 30C

- Use **delimiter=** to set the delimiting character to display between different headers and fields. This delimiter must be a single character. By default, the delimiter is a space.

Output customization applies only to the output for certain **lshosts** options:

- **LSF_LSHOSTS_FORMAT** and **lshosts -o** both apply to output for the **lshosts** command with no options, and for **lshosts** options with output that filter information, including the following options: **-a**, **-cname**.
- **LSF_LSHOSTS_FORMAT** and **lshosts -o** do not apply to output for other **lshosts** options that use a modified format, including the following options: **-l**, **-w**.

The **lshosts -o** option overrides the **LSF_LSHOSTS_FORMAT** environment variable, which overrides the **LSF_LSHOSTS_FORMAT** setting in **lsf.conf**.

Valid values

The following are the field names used to specify the **lshosts** fields to display, recommended width, aliases you can use instead of field names, and units of measurement for the displayed field:

Field name	Width	Aliases	Unit
HOST_NAME	20	hname	
type	10		
model	10		
cpuf	10		
max	10		
ncpus	8		
maxmem	10		LSF_UNIT_FOR_LIMITS in lsf.conf (KB by default)
maxswp	10		LSF_UNIT_FOR_LIMITS in lsf.conf (KB by default)
server	10		
RESOURCES	20	res	
ndisks	8		

Table 7. Output fields for `lshosts` (continued)

Field name	Width	Aliases	Unit
maxtmp	10		LSF_UNIT_FOR_LIMITS in lsf.conf (KB by default)
rexpri	10		
nprocs	8		
ncores	8		
nthreads	10		
RUN_WINDOWS	20	runwin	

Field names and aliases are case-sensitive. Valid values for the output width are any positive integer 1 - 4096.

Example

```
LSF_LSHOSTS_FORMAT="HOST_NAME type: RESOURCES:- RUN_WINDOWS:-16 delimiter='^'"
```

Running **lshosts** displays the following fields:

- HOST_NAME with unlimited width and left-aligned.
- type with a maximum width of 10 characters (which is the recommended width) and left-aligned.
- RESOURCES with a maximum width of 20 characters (which is the recommended width) and right-aligned.
- RUN_WINDOWS with a maximum width of 16 characters and right-aligned.
- The ^ character is displayed between different headers and fields.

Default

Not defined. The current **lshosts** output is used.

LSF_LSLOAD_FORMAT

Syntax

```
LSF_LSLOAD_FORMAT="field_name[:[:-][output_width]][:unit]] ... [delimiter='character']"
```

Description

Sets the customized output format for the **lsload** command.

- Specify which **lsload** fields, in which order, and with what width to display.
- Specify the asterisk wildcard character (*****) in the field name to specify multiple external resource names. You can only specify one asterisk, but this asterisk can be at any position in the field name.

For example, running **lsload -o "gpu_mode*"** shows fields such as `gpu_mode0`, `gpu_mode1`, `gpu_mode2`, `gpu_mode10`, `gpu_mode11`, and `gpu_mode12`.

- Specify only the **lsload** field name to set its output to unlimited width and left justification.
- Specify the width colon (:) without a width to set the output width to the recommended width for that field.
- Specify the width colon (:) with a width to set the maximum number of characters to display for the field. When its value exceeds this width, **lsload** truncates the ending characters.

- Specify a hyphen (-) to set right justification when **lsload** displays the output for the specific field. If not specified, the default is to set left justification when **lsload** displays the output for a field.
- Specify the unit colon (:) with a unit to set the unit for the output of the specific field:
 - Specify S to use a built-in conversion for space or capacity, such as memory or disk space. Values are automatically scaled for M (MB), G (GB), and T (TB), where the default unit is M (MB).

For example, when displaying the **mem** field with a specified width of 3,

- For a value of 30, running the **lsload -o "mem:3:S"** command shows 30.0M.
- For a value of 4096, running the **lsload -o "mem:3:S"** command shows 4.0G.
- For a value of 5000000, running the **lsload -o "mem:3:S"** command shows 4.8T.
- Specify D to use a built-in conversion for duration or time, such as memory or disk space. Values are automatically scaled for s (seconds), m (minutes), h (hours), and d (days), where the default unit is s (seconds). The automatically scaled value is rounded up after the first decimal point.

For example, when displaying the external **mytime** resource field with a specified width of 5,

- For a value of 30, running the **lsload -o "mytime:5:D"** command shows 30.0s.
- For a value of 8000, running the **lsload -o "mytime:5:D"** command shows 2.2h.
- For a value of 5000000, running the **lsload -o "mytime:5:D"** command shows 57.8d.
- Specify any other string of 1 - 3 characters and the characters are used as is in the field value. The first character must be a letter (upper or lower case). The second and third characters must be an alphanumeric character.

For example, when displaying the external **gpu_temp** resource with a width of 3, running the **lsload -o "gpu_temp:3:C"** command for a value of 30 shows 30C

- Use `delimiter=` to set the delimiting character to display between different headers and fields. This delimiter must be a single character. By default, the delimiter is a space.

Output customization applies only to the output for certain **lsload** options:

- **LSF_LSLOAD_FORMAT** and **lsload -o** both apply to output for the **lsload** command with no options, and for **lsload** options with short form output that filter information, including the following options: -a, -E, -cname, -N, -n, -R.
- **LSF_LSLOAD_FORMAT** and **lsload -o** do not apply to output for **lsload** options that use a modified format, including the following options: -I, -l, -w, -s.

The **lsload -o** option overrides the **LSF_LSLOAD_FORMAT** environment variable, which overrides the **LSF_LSLOAD_FORMAT** setting in `lsf.conf`.

Valid values

By default, the **lsload** command displays the built-in resource indices. You can also specify the names of external resources. The following are the field names for the built-in resource indices that are used to specify the **lsload** fields to display, recommended width, and units of measurement for the displayed field:

Field name	Width	Unit
HOST_NAME	20	
status	15	
r15s	6	
r1m	6	
r15m	6	

Table 8. Output fields for `lsload` (continued)

Field name	Width	Unit
ut	6	
pg	6	
ls	6	
it	6	
io	6	
tmp	10	LSF_UNIT_FOR_LIMITS in <code>lsf.conf</code> (KB by default)
swp	10	LSF_UNIT_FOR_LIMITS in <code>lsf.conf</code> (KB by default)
mem	10	LSF_UNIT_FOR_LIMITS in <code>lsf.conf</code> (KB by default)
gpu_status* For example, <code>gpu_status0</code> and <code>gpu_status1</code> if there are two GPUs.	10	
gpu_error* For example, <code>gpu_error0</code> and <code>gpu_error1</code> if there are two GPUs.	20	

Field names are case-sensitive. Valid values for the output width are any positive integer 1 - 4096.

Example

```
LSF_LSLOAD_FORMAT="HOST_NAME status: r15s:- r1m:7 r15m:-8 tmp:S swp::S mem:9:S delimiter='^'"
```

Running **lsload** displays the following fields:

- `HOST_NAME` with unlimited width and left-aligned.
- `status` with a maximum width of 15 characters (which is the recommended width) and left-aligned.
- `r15s` with a maximum width of 6 characters (which is the recommended width) and right-aligned.
- `r1m` with a maximum width of 7 characters and left-aligned.
- `r15m` with a maximum width of 8 characters and right-aligned.
- `tmp` with unlimited width, left-aligned, and automatically scaled for space or capacity (MB, GB, and TB).
- `swp` with a maximum width of 10 characters (which is the recommended width), left-aligned, and automatically scaled for space or capacity (MB, GB, and TB)
- `mem` with a maximum width of 9 characters, left-aligned, and automatically scaled for space or capacity (MB, GB, and TB)
- The `^` character is displayed between different headers and fields.

Default

Not defined. The current **lsload** output is used.

LSF_LSLOGIN_SSH

Enables SSH to secure communication between hosts and during job submission.

Syntax

LSF_LSLOGIN_SSH=Y | y

Description

A user with SSH configured can log on to a remote host without providing a password. All communication between local and remote hosts is encrypted.

SSH is used when running any of the following:

- Remote login to a lightly loaded host (**lslogin**)
- An interactive job (**bsub -IS | -ISp | ISs**)
- An X-window job (**bsub -IX**)
- An externally submitted job that is interactive or X-window job (**esub**)

Default

Not set. LSF uses **rlogin** to authenticate users.

LSF_MACHDEP

Syntax

LSF_MACHDEP=*directory*

Description

Specifies the directory in which machine-dependent files are installed. These files cannot be shared across different types of machines.

In clusters with a single host type, LSF_MACHDEP is usually the same as LSF_INDEP. The machine dependent files are the user commands, daemons, and libraries. You should not need to modify this parameter.

As shown in the following list, LSF_MACHDEP is incorporated into other LSF variables.

- LSF_BINDIR=\$LSF_MACHDEP/bin
- LSF_LIBDIR=\$LSF_MACHDEP/lib
- LSF_SERVERDIR=\$LSF_MACHDEP/etc
- XLSF_UIDDIR=\$LSF_MACHDEP/lib/uid

Default

/usr/share/lsf

See also

LSF_INDEP

LSF_MANAGE_FREQUENCY

Syntax

LSF_MANAGE_FREQUENCY=N | CORE | HOST

Description

Uses a keyword value (N, CORE, or HOST) to set whether the CPU frequency is set for the core (CPU) or by host (node). If the value CORE is set, jobs will require affinity resource requirements. The default value for this parameter is N (not set).

Default

N

LSF_MANDIR**Syntax**

LSF_MANDIR=*directory*

Description

Directory under which all man pages are installed.

The man pages are placed in the man1, man3, man5, and man8 subdirectories of the LSF_MANDIR directory. This is created by the LSF installation process, and you should not need to modify this parameter.

Man pages are installed in a format suitable for BSD-style **man** commands.

For most versions of UNIX and Linux, you should add the directory LSF_MANDIR to your MANPATH environment variable. If your system has a **man** command that does not understand MANPATH, you should either install the man pages in the `/usr/man` directory or get one of the freely available **man** programs.

Default

LSF_INDEP/man

LSF_MASTER_LIST**Syntax**

LSF_MASTER_LIST="*host_name ...*"

Description

Required. Defines a list of hosts that are candidates to become the master host for the cluster.

Listed hosts must be defined in `lsf.cluster.cluster_name`.

Host names are separated by spaces.

Tip:

On UNIX and Linux, master host candidates should share LSF configuration and binaries. On Windows, configuration files are shared, but not binaries.

Starting in LSF 7, LSF_MASTER_LIST *must* be defined in `lsf.conf`.

If EGO is enabled, LSF_MASTER_LIST can only be defined in `lsf.conf`. EGO_MASTER_LIST can only be defined in `ego.conf`. EGO_MASTER_LIST cannot be defined in `lsf.conf`. LSF_MASTER_LIST cannot be defined in `ego.conf`.

LIM reads EGO_MASTER_LIST wherever it is defined. If both LSF_MASTER_LIST and EGO_MASTER_LIST are defined, the value of EGO_MASTER_LIST in `ego.conf` is taken. To avoid errors, you should make

sure that the value of LSF_MASTER_LIST matches the value of EGO_MASTER_LIST, or define only EGO_MASTER_LIST.

If EGO is disabled, `ego.conf` not loaded and the value of LSF_MASTER_LIST defined in `lsf.conf` is taken.

When you run **lsadmin reconfig** to reconfigure the cluster, only the master LIM candidates read `lsf.shared` and `lsf.cluster.cluster_name` to get updated information. The elected master LIM sends configuration information to slave LIMs.

If you have a large number of non-master hosts, you should configure LSF_LIM_IGNORE_CHECKSUM=Y to ignore warning messages like the following logged to lim log files on non-master hosts.

```
Feb 26 13:47:35 2013 9746 4 10.1 xdr_loadvector:
Sender <10.225.36.46:9999> has a different configuration
```

Interaction with LSF_SERVER_HOSTS

You can use the same list of hosts, or a subset of the master host list defined in LSF_MASTER_LIST, in LSF_SERVER_HOSTS. If you include the primary master host in LSF_SERVER_HOSTS, you should define it as the last host of the list.

If LSF_ADD_CLIENTS is defined in `install.config` at installation, **lsfinstall** automatically appends the hosts in LSF_MASTER_LIST to the list of hosts in LSF_SERVER_HOSTS so that the primary master host is last. For example:

```
LSF_MASTER_LIST="lsfmaster hostE"
```

```
LSF_SERVER_HOSTS="hostB hostC hostD hostE lsfmaster"
```

The value of LSF_SERVER_HOSTS is not changed during upgrade.

EGO parameter

EGO_MASTER_LIST

Default

Defined at installation

See also

LSF_LIM_IGNORE_CHECKSUM

LSF_MASTER_NSLOOKUP_TIMEOUT

Syntax

```
LSF_MASTER_NSLOOKUP_TIMEOUT=time_milliseconds
```

Description

Timeout in milliseconds that the master LIM waits for DNS host name lookup.

If LIM spends a lot of time calling DNS to look up a host name, LIM appears to hang.

This parameter is used by master LIM only. Only the master LIM detects this parameter and enable the DNS lookup timeout.

Default

Not defined. No timeout for DNS lookup

See also**LSF_LIM_IGNORE_CHECKSUM****LSF_MAX_TRY_ADD_HOST****Syntax**LSF_MAX_TRY_ADD_HOST=*integer***Description**

When a slave LIM on a dynamically added host sends an add host request to the master LIM, but master LIM cannot add the host for some reason, the slave LIM tries again. LSF_MAX_TRY_ADD_HOST specifies how many times the slave LIM retries the add host request before giving up.

Default

20

LSF_MC_FORWARD_FAIRSHARE_CHARGE_DURATION**Syntax**LSF_MC_FORWARD_FAIRSHARE_CHARGE_DURATION=*seconds***Description**

For LSF multicluster capability, specifies the duration of time after which LSF removes the forwarded jobs from the user priority calculation for fairshare scheduling. If the global fairshare daemon is enabled, set this parameter to 300 seconds (5 minutes), or enough time to reflect the job usage from the remote cluster on the fairshare counter.

Default

Not defined.

LSF_MC_NON_PRIVILEGED_PORTS**Syntax**LSF_MC_NON_PRIVILEGED_PORTS=*y | Y***Description**

MultiCluster only. If this parameter is enabled in one cluster, it must be enabled in all clusters. Specify Y to make LSF daemons use non-privileged ports for communication across clusters.

Compatibility

This disables privileged port daemon authentication, which is a security feature. If security is a concern, you should use **auth** for LSF daemon authentication (see LSF_AUTH_DAEMONS in `lsf.conf`).

Default

Not defined. LSF daemons use privileged port authentication

LSF_MISC

Syntax

LSF_MISC=*directory*

Description

Directory in which miscellaneous machine independent files, such as example source programs and scripts, are installed.

Default

LSF_CONFDIR/misc

LSF_MQ_BROKER_HOSTS

For LSF resource connector. Enables support for the **bhosts -rc** and **bhosts -rconly** command options to get LSF resource connector provider host information

Syntax

LSF_MQ_BROKER_HOSTS=*host_name*

Description

If you use the MQTT message broker daemon (**mosquitto**) that is provided with LSF resource connector, specifies the host name where LIM starts that **mosquitto** daemon. The MQTT message broker receives provider host information from **ebrokerd** and publishes that information for the **bhosts -rc** and **bhosts -rconly** command options to display. If the **LSF_MQ_BROKER_HOSTS** parameter is not specified, the MQTT broker is not started, and the **bhosts -rc** and **bhosts -rconly** command options fail.

Specify a host that runs the MQTT message broker daemon (**mosquitto**). When LIM starts, it checks the **LSF_MQ_BROKER_HOSTS** parameter. LIM manages the message broker lifecycle, terminating and restarting it when necessary.

Failure of the **mosquitto** daemon is recorded in the LIM log.

The *host_name* is one of the existing LSF cluster hosts. After configuring this parameter, run the **lsadmin limrestart** command on the master host. If the specified host name is not the master, run the **lsadmin limrestart host_name** command.

To verify that the **mosquitto** daemon is up and running, use the `ps -ef | grep mosquitto` command.

Example

```
LSF_MQ_BROKER_HOSTS=hosta
```

Default

Not defined.

See also

- **EBROKERD_HOST_CLEAN_DELAY**
- **MQTT_BROKER_HOST**
- **MQTT_BROKER_PORT**
- **bhosts -rc** and **bhosts -rconly**

LSF_NIOS_DEBUG

Syntax

LSF_NIOS_DEBUG=1

Description

Enables NIOS debugging for interactive jobs (if LSF_NIOS_DEBUG=1).

NIOS debug messages are written to standard error.

This parameter can also be defined as an environment variable.

When LSF_NIOS_DEBUG and LSF_CMD_LOGDIR are defined, NIOS debug messages are logged in `nios.log.host_name` in the location specified by LSF_CMD_LOGDIR.

If LSF_NIOS_DEBUG is defined, and the directory defined by LSF_CMD_LOGDIR is inaccessible, NIOS debug messages are logged to `/tmp/nios.log.host_name` instead of `stderr`.

On Windows, NIOS debug messages are also logged to the temporary directory.

Default

Not defined

See also

LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_NIOS_ERR_LOGDIR

Syntax

LSF_NIOS_ERR_LOGDIR=*directory*

Description

Applies to Windows only.

If LSF_NIOS_ERR_LOGDIR is specified, logs NIOS errors to *directory/nios.error.log.hostname.txt*.

If the attempt fails, LSF tries to write to another directory instead. The order is:

1. the specified log directory
2. LSF_TMPDIR
3. %TMP%
4. %TEMP%
5. the system directory, for example, C:\winnt

If LSF_NIOS_DEBUG is also specified, NIOS debugging overrides the LSF_NIOS_ERR_LOGDIR setting.

LSF_NIOS_ERR_LOGDIR is an alternative to using the NIOS debug functionality.

This parameter can also be defined as an environment variable.

Default

Not defined

See also

LSF_NIOS_DEBUG, **LSF_CMD_LOGDIR**

LSF_NIOS_JOBSTATUS_INTERVAL

Syntax

`LSF_NIOS_JOBSTATUS_INTERVAL=time_minutes`

Description

Time interval at which NIOS polls mbatchd to check if a job is still running. Applies to interactive batch jobs and blocking jobs.

Use this parameter if you have scripts that depend on an exit code being returned.

If this parameter is not defined and a network connection is lost, mbatchd cannot communicate with NIOS and the return code of a job is not retrieved.

When **LSF_NIOS_JOBSTATUS_INTERVAL** is defined, NIOS polls mbatchd on the defined interval to check if a job is still running (or pending). NIOS continues to poll mbatchd until it receives an exit code or mbatchd responds that the job does not exist (if the job has already been cleaned from memory for example).

For interactive jobs NIOS polls mbatchd to retrieve a job's exit status when this parameter is enabled and:

- the connection between NIOS and the job RES is broken. For example, a network failure between submission host and execution host occurs.
- job RES runs abnormally. For example, it is out of memory.
- job is waiting for dispatch.

For blocking jobs, NIOS will always poll mbatchd to retrieve a job's exit status when this parameter is enabled,

If an exit code cannot be retrieved, NIOS generates an error message and the code -11.

Valid values

Any integer greater than zero.

Default

Not defined

Notes

Set this parameter to large intervals such as 15 minutes or more so that performance is not negatively affected if interactive jobs are pending for too long. NIOS always calls **mbatchd** on the defined interval to confirm that a job is still pending and this may add load to **mbatchd**.

See also

Environment variable **LSF_NIOS_PEND_TIMEOUT**

LSF_NIOS_MAX_TASKS

Syntax

`LSF_NIOS_MAX_TASKS=integer`

Description

Specifies the maximum number of NIOS tasks.

Default

Not defined

LSF_NIOS_PORT_RANGE**Syntax**

LSF_NIOS_PORT_RANGE="*min_port_number-max_port_number*"

Description

Defines a range of listening ports for NIOS to use.

If not defined, LSF assigns a random NIOS port number.

Default

Not defined.

LSF_NIOS_RES_HEARTBEAT**Syntax**

LSF_NIOS_RES_HEARTBEAT=*time_minutes*

Description

Applies only to interactive non-parallel batch jobs.

Defines how long NIOS waits before sending a message to RES to determine if the connection is still open.

Use this parameter to ensure NIOS exits when a network failure occurs instead of waiting indefinitely for notification that a job has been completed. When a network connection is lost, RES cannot communicate with NIOS and as a result, NIOS does not exit.

When this parameter is defined, if there has been no communication between RES and NIOS for the defined period of time, NIOS sends a message to RES to see if the connection is still open. If the connection is no longer available, NIOS exits.

Valid values

Any integer greater than zero

Default

Not defined

Notes

The time you set this parameter to depends how long you want to allow NIOS to wait before exiting. Typically, it can be a number of hours or days. Too low a number may add load to the system.

LSF_NON_PRIVILEGED_PORTS

Syntax

LSF_NON_PRIVILEGED_PORTS=Y | N

Description

By default, LSF uses non privileged ports for communication (**LSF_NON_PRIVILEGED_PORTS=Y**).

If **LSF_NON_PRIVILEGED_PORTS=N**, LSF clients (LSF commands and daemons) do not use privileged ports to communicate with daemons and LSF daemons do not check privileged ports for incoming requests to do authentication.

To use privileged port communication, change the value of **LSF_NON_PRIVILEGED_PORTS** to N, then do the following:

1. Shut down the cluster.
2. Change the parameter value.
3. Restart the cluster so the new value takes effect.

For migration and compatibility for each cluster:

- If all hosts were upgraded to LSF 9.1 or later, they work with non-privileged ports.
- If all hosts were upgraded to LSF 9.1 or later, but you want to use privileged ports for communication, then set **LSF_NON_PRIVILEGED_PORTS** to N and make sure that the value for **LSB_MAX_JOB_DISPATCH_PER_SESSION** is less than 300.
- If the master host is upgraded to LSF 9.1 or later, but some server hosts are still running older versions, and if the value defined for **LSB_MAX_JOB_DISPATCH_PER_SESSION** is above 300, then no changes are required. If the value is less than 300, then you need to set **LSF_NON_PRIVILEGED_PORTS=Y**. This tells the old **sbatchd** to use non-privileged ports for communication.

Default

The default value is Y, which means LSF uses non-privileged port communication.

LSF_PAM_APPL_CHKPNT

Syntax

LSF_PAM_APPL_CHKPNT=Y | N

Description

When set to Y, allows PAM to function together with application checkpointing support.

Default

Y

LSF_PAM_CLEAN_JOB_DELAY

Syntax

LSF_PAM_CLEAN_JOB_DELAY=*time_seconds*

Description

The number of seconds LSF waits before killing a parallel job with failed tasks. Specifying `LSF_PAM_CLEAN_JOB_DELAY` implies that if any parallel tasks fail, the entire job should exit without running the other tasks in the job. The job is killed if any task exits with a non-zero exit code.

Specify a value greater than or equal to zero (0).

Applies only to PAM jobs.

Default

Undefined: LSF kills the job immediately

LSF_PAM_HOSTLIST_USE**Syntax**

`LSF_PAM_HOSTLIST_USE=unique`

Description

Used to start applications that use both OpenMP and MPI.

Valid values

```
unique
```

Default

Not defined

Notes

At job submission, LSF reserves the correct number of processors and PAM starts only 1 process per host. For example, to reserve 32 processors and run on 4 processes per host, resulting in the use of 8 hosts:

```
bsub -n 32 -R "span[ptile=4]" pam yourOpenMPJob
```

Where defined

This parameter can alternatively be set as an environment variable. For example:

```
setenv LSF_PAM_HOSTLIST_USE unique
```

LSF_PAM_PLUGINDIR**Syntax**

`LSF_PAM_PLUGINDIR=path`

Description

The path to `libpamvc1.so`. Used with LSF HPC features.

Default

Path to `LSF_LIBDIR`

LSF_PAM_USE_ASH

Syntax

LSF_PAM_USE_ASH=y | Y

Description

Enables LSF to use the SGI IRIX Array Session Handles (ASH) to propagate signals to the parallel jobs.

See the IRIX system documentation and the `array_session(5)` man page for more information about array sessions.

Default

Not defined

LSF_PASSWD_DIR

Syntax

LSF_PASSWD_DIR=*file_path*

Description

Defines a location for LSF to load and update the `passwd.lsfuser` file, used for registering `lspassword` for a Windows user account.

Note: `LSF_PASSWD_DIR` does not need to be configured if the cluster contains no Windows users.

Note: `passwd.lsfuser` is automatically generated by LSF - it does not need to be created.

Specify the full path to a shared directory accessible by all master candidate hosts. The LSF `lim` daemon must have read and write permissions on this directory.

By default, `passwd.lsfuser` is located in `$LSF_CONFDIR`. The default location is only used if `LSF_PASSWD_DIR` is undefined; if you define a new location and `lim` fails to access `passwd.lsfuser` in `LSF_PASSWD_DIR`, it will not check `$LSF_CONFDIR`.

You must restart `lim` to make changes take effect.

Default

Not defined (that is, `passwd.lsfuser` is located in `$LSF_CONFDIR`)

LSF_PE_NETWORK_NUM

Syntax

LSF_PE_NETWORK_NUM=*num_networks*

Description

For LSF IBM Parallel Environment (PE) integration. Specify a value between 0 and 8 to set the number of InfiniBand networks on the host. If the number is changed, run `lsadmin reconfig` and `badmin mbdrestart` to make the change take effect

`LSF_PE_NETWORK_NUM` must be defined with a non-zero value in `lsf.conf` for LSF to collect network information to run PE jobs.

Note: If `LSF_PE_NETWORK_NUM` is configured with a valid value, `MAX_JOBID` in `lsb.params` should not be configured with a value larger than 4194303 and `MAX_JOB_ARRAY_SIZE` in `lsb.params` should not

be configured with a value larger than 1023. Otherwise, the jobID may not be represented correctly in PE and jobs do not run as expected.

Example

For example, hostA has two networks: 18338657685884897280 and 18338657685884897536. Each network has 256 windows. Set LSF_PE_NETWORK_NUM=2.

Maximum

8

Default

0

LSF_PE_NETWORK_UPDATE_INTERVAL

Syntax

LSF_PE_NETWORK_UPDATE_INTERVAL=*seconds*

Description

For LSF IBM Parallel Environment (PE) integration. When LSF collects network information for PE jobs, LSF_PE_NETWORK_UPDATE_INTERVAL specifies the interval for updating network information.

Default

15 seconds

LSF_PIM_INFODIR

Syntax

LSF_PIM_INFODIR=*path*

Description

The path to where PIM writes the `pim.info.host_name` file.

Specifies the path to where the process information is stored. The process information resides in the file `pim.info.host_name`. The PIM also reads this file when it starts so that it can accumulate the resource usage of dead processes for existing process groups.

EGO parameter

EGO_PIM_INFODIR

Default

Not defined. The system uses `/tmp`.

LSF_PIM_LINUX_ENHANCE

Syntax

LSF_PIM_LINUX_ENHANCE=Y | N

Description

When enabled, the PIM daemon reports proportional memory utilization for each process attached to a shared memory segment. The sum total of memory utilization of all processes on the host is now accurately reflected in the total memory used. (The Linux kernel must be version 2.6.14 or newer.)

When **EGO_PIM_SWAP_REPORT** is set, the swap amount is correctly reported. The swap amount is the virtual memory minus the value of the rss value in the static Linux file.

Applies only to Linux operating systems and Red Hat Enterprise Linux 4.7.5.0.

Default

Set to Y at time of installation. If otherwise undefined, then N.

LSF_PIM_SLEEPTIME**Syntax**

LSF_PIM_SLEEPTIME=*time_seconds*

Description

The reporting period for PIM.

PIM updates the process information every 15 minutes unless an application queries this information. If an application requests the information, PIM updates the process information every LSF_PIM_SLEEPTIME seconds. If the information is not queried by any application for more than 5 minutes, the PIM reverts back to the 15 minute update period.

EGO parameter

EGO_PIM_SLEEPTIME

Default

30 seconds

LSF_PIM_SLEEPTIME_UPDATE**Syntax**

LSF_PIM_SLEEPTIME_UPDATE=y | n

Description

UNIX only.

Use this parameter to improve job throughput and reduce a job's start time if there are many jobs running simultaneously on a host. This parameter reduces communication traffic between sbatchd and PIM on the same host.

When this parameter is not defined or set to n, sbatchd queries PIM as needed for job process information.

When this parameter is defined, sbatchd does not query PIM immediately as it needs information; sbatchd only queries PIM every LSF_PIM_SLEEPTIME seconds.

Limitations

When this parameter is defined:

- sbatchd may be intermittently unable to retrieve process information for jobs whose run time is smaller than LSF_PIM_SLEEPTIME.
- It may take longer to view resource usage with **bjobs -l**.

EGO parameter

EGO_PIM_SLEEPTIME_UPDATE

Default

Set to y at time of installation. Otherwise, not defined.

LSF_PLATFORM_COMPATIBILITY

Allows for backward compatibility of the Platform name after LSF 10.1.

Syntax

LSF_PLATFORM_COMPATIBILITY=Y|y|N|n

Description

Set it to y|Y in `lsf.conf` to enable `lsid` and the LSF command **-V** to display "IBM Platform LSF" instead of "IBM Spectrum LSF". This solves compatibility issues between LSF10.1 and older versions of PPM.

Default

Not defined

LSF_POE_TIMEOUT_BIND**Syntax**

LSF_POE_TIMEOUT_BIND=*time_seconds*

Description

This parameter applies to the PAM Taskstarter. It specifies the time in seconds for the **poe_w** wrapper to keep trying to set up a server socket to listen on.

`poe_w` is the wrapper for the IBM **poe** driver program.

LSF_POE_TIMEOUT_BIND can also be set as an environment variable for `poe_w` to read.

Default

120 seconds

LSF_POE_TIMEOUT_SELECT**Syntax**

LSF_POE_TIMEOUT_SELECT=*time_seconds*

Description

This parameter applies to the PAM Taskstarter. It specifies the time in seconds for the `poe_w` wrapper to wait for connections from the `pmd_w` wrapper. `pmd_w` is the wrapper for **pmd** (IBM PE Partition Manager Daemon).

LSF_POE_TIMEOUT_SELECT can also be set as an environment variable for `poe_w` to read.

Default

160 seconds

LSF_PROCESS_TRACKING**Syntax**

LSF_PROCESS_TRACKING=Y|y|N|n

Description

Enable this parameter to track processes based on job control functions such as termination, suspension, resume and other signaling, on Linux systems which support cgroups' freezer subsystem. Once enabled, this parameter takes effect for new jobs.

Disable this parameter if you want LSF to depend on PIM's updates for tracking the relationship between jobs and process.

Default

Set to Y at time of installation. If otherwise undefined, then N.

LSF_QUERY_ES_FUNCTIONS**Syntax**

LSF_QUERY_ES_FUNCTIONS="all"

LSF_QUERY_ES_FUNCTIONS="[acct] |[jobs] |[energy] |[gpfsio]"

Description

Specifies the functions that are permitted to use IBM Spectrum LSF Explorer (LSF Explorer) to access and retrieve additional data. Specify a space-separated list of function names.

The **acct** and **jobs** functions integrate **bacct** and **bhist** with LSF Explorer Server. Using LSF Explorer to get data improves the performance of the supported commands by avoiding the need for parsing large log files whenever you run these commands.

This **energy** function integrates LSF Explorer energy accounting into LSF. LSF uses LSF Explorer to collect energy data from each host by the LSF Beats services (**lsfbeat** tool). The **bjobs** and **bhosts** commands can then query the data from LSF Explorer and display the job level or host level energy to users.

To use this parameter, you must first define the **LSF_QUERY_ES_SERVERS** parameter to enable LSF to use LSF Explorer to retrieve the data.

Valid values**acct**

If the **acct** function is defined, the **bacct** command uses LSF Explorer to retrieve the accounting log records.

jobs

If the **jobs** function is defined, the **bhist** command uses LSF Explorer to retrieve the event log records.

energy

If the **energy** function is defined and **LSF_ENABLE_BEAT_SERVICE="energy"** is defined in the **lsf.conf** file, the **bhosts** and **bjobs** commands use LSF Explorer to retrieve energy data.

gpfsio

If the **gpfsio** function is defined, the **bjobs** and **bacct** commands use LSF Explorer to retrieve IBM Spectrum Scale job accounting I/O data.

Specifically, the **bjobs -l** command shows approximate accumulated job disk usage (I/O) data, the **bjobs -o "gpfsio"** command shows job disk usage (I/O) data, and the **bacct -l** command shows the total approximate number of read/write bytes of all storage pools, all on IBM Spectrum Scale.

all

If the **all** keyword is defined, all supported functions use LSF Explorer to retrieve data.

Default

Not defined. The commands do not use LSF Explorer to retrieve data.

See also

LSF_QUERY_ES_SERVERS, **LSF_ENABLE_BEAT_SERVICE**

LSF_QUERY_ES_SERVERS

Syntax

```
LSF_QUERY_ES_SERVERS="[ip_address:port_number | host_name:port_number] ..."
```

Description

To use this parameter, IBM Spectrum LSF Explorer (LSF Explorer), Version 10.2, or later, must be installed and working.

Use this parameter to configure the IP address and port number of the LSF Explorer Server, and to enable the supported commands to use the LSF Explorer Elasticsearch features.

Use a space-separated list of IP addresses and port numbers or host names and port numbers of the LSF Explorer Servers. If the first server does not respond, LSF attempts to contact the subsequent servers to retrieve data. For example,

```
LSF_QUERY_ES_SERVERS="192.168.0.1:8080 hostE:8080"
```

Use the **LSF_QUERY_ES_FUNCTIONS** parameter to define which commands are permitted to use LSF Explorer to retrieve data.

Default

Not defined. The commands do not use LSF Explorer to retrieve data.

See also

LSF_QUERY_ES_FUNCTIONS, **LSF_ENABLE_BEAT_SERVICE**

LSF_RC_AUDIT_LOG

Syntax

```
LSF_RC_AUDIT_LOG=Y | y | N | n
```

Description

If set to Y or y, enables the LSF resource connector auditor to generate log files.

Default

N.

See also

RC_MAX_AUDIT_LOG_SIZE, RC_MAX_AUDIT_LOG_KEEP_TIME

LSF_REG_FLOAT_HOSTS**Syntax**

LSF_REG_FLOAT_HOSTS=Y|y|N|n

Description

Enables registration handling for LSF hosts without requiring the use of DNS to resolve local hosts.

If set to Y or y on the LSF server, allows LIM on the server to receive registration requests from hosts that are running the **lsreghost** service. LIM then stores the host name, IP address, and (on Windows only) the computer SID of the host in the `reghostscache` file. LSF daemons look for host records in this file when attempting to look up host names or IP addresses before using DNS.

By default, the `reghostscache` file is stored in the file path as defined by the **LSB_SHAREDIR** parameter in the `lsf.conf` file. Define the **LSB_SHAREDIR** parameter so that the `reghostscache` file can be shared with as many LSF servers as possible. For all LSF servers that have access to the shared directory defined by **LSB_SHAREDIR**, only one of these servers need to receive the registration request from the host. This reduces network load by reducing the number of servers to which the registration request must be sent. If all hosts in the cluster can access **LSB_SHAREDIR**, the registration only needs to be sent to the master LIM, which records the host information in the shared `reghostscache` file that all other servers can access. If **LSB_SHAREDIR** is not defined, the `reghostscache` file is placed in the **LSF_TOP** directory.

After enabling this parameter on the LSF servers, create the `hostregsetup` file on the local host or a location that is accessible to the local host. This file contains a list of LSF servers to which the local host must register itself, with one server name on each line (empty lines and `#comment` text are ignored).

Note: If **LSF_SHAREDIR** is accessible to all servers in the cluster, only the master LIM host needs to be specified in the `hostregsetup` file.

Run the **lsreghost** command with the full file path to the `hostregsetup` file to enable the local to register itself to the specified LSF servers:

- On a Windows host, run `lsreghost -i file_path\hostregsetup` to install the **lsreghost** service onto the local host, which runs as a service that automatically starts up every time the local machine starts up. For example, `lsreghost -i C:\LSF_9.1\hostregsetup`
- On a Linux host, run `lsreghost -s file_path\hostregsetup` with root privileges to have the local host send a single registration message to all LSF servers that are listed in the `hostregsetup` file. If you want to run **lsreghost** at set intervals, set up a Linux **cron** job.

For details on all **lsreghost** command options, run **lsreghost -h** on the local host.

If **LSF_CONNECTION_CHANGE=Y** is set in the `lsf.conf` file on the local host, **lsreghost** also registers the local host with the LSF servers if it detects a change in the total number of connections (that is, IP addresses) that are associated with the local host.

If the **LSF_REG_HOST_INTERVAL** parameter is set to an integer in the `lsf.conf` file on the local host, **lsreghost** sends a new registration message to the LSF servers after the defined interval.

Default

N

See also

LSF_CONNECTION_CHANGE, LSF_REG_HOST_INTERVAL.

LSF_REG_HOST_INTERVAL

Syntax

LSF_REG_HOST_INTERVAL=*integer*

Description

Windows only. Specifies the interval, in minutes, in which **lsreghost** sends additional registration messages to LSF servers.

This parameter is only valid if registration handling is enabled for LSF hosts (that is, LSF_REG_FLOAT_HOSTS=Y is set in the `lsf.conf` file on the LSF server).

Specify an integer for this parameter to set the time interval, in minutes, for **lsreghost** to send registration messages to LSF servers.

You must restart the **lsreghost** command on the local host for this change to take effect.

Default

N

See also

LSF_REG_FLOAT_HOSTS, LSF_CONNECTION_CHANGE.

LSF_REMOTE_COPY_CMD

Syntax

LSF_REMOTE_COPY_CMD="*copy_command*"

Description

UNIX only. Specifies the shell command or script to use with the following LSF commands if RES fails to copy the file between hosts.

- **lsrcp**
- **bsub -i, -f, -is, -Zs** "Ci(s)
- **bmod -Zs**

By default, **rcp** is used for these commands.

There is no need to restart any daemons when this parameter changes.

For example, to use **scp** instead of **rcp** for remote file copying, specify:

```
LSF_REMOTE_COPY_CMD="scp -B -o 'StrictHostKeyChecking no'"
```

You can also configure **ssh** options such as BatchMode, StrictHostKeyChecking in the global `SSH_ETC/ssh_config` file or `$HOME/.ssh/config`.

When remote copy of a file via RES fails, the environment variable "LSF_LSRCP_ERRNO" is set to the system defined errno. You can use this variable in a self-defined shell script executed by **lsrcp**. The script can do the appropriate cleanup, recopy, or retry, or it can just exit without invoking any other copy command.

LSF automatically appends two parameters before executing the command:

- The first parameter is the source file path.
- The second parameter is the destination file path.

Valid values

Values are passed directly through. Any valid scp, rcp, or custom copy commands and options are supported except for compound multi-commands. For example, set **LSF_REMOTE_COPY_CMD**="scp -B -o 'StrictHostKeyChecking no'".

To avoid a recursive loop, the value of **LSF_REMOTE_COPY_CMD** must not be **lsrcp** or a shell script executing **lsrcp**.

Default

Not defined.

LSF_REPLACE_PIM_WITH_LINUX_CGROUP

Minimizes the impact of PIM daemon processing overhead on parallel jobs.

Syntax

LSF_REPLACE_PIM_WITH_LINUX_CGROUP=y|Y|n|N

Description

PIM collects job processes, the relationship among all processes, the memory usage of each process, and the CPU time of each process periodically. Those actions can influence the execution of parallel jobs (so-called *OS jitter*). To minimize OS jitter, you can configure the LSF cgroup feature. Linux cgroups function similarly to PIM, but can generate less OS jitter than PIM. The

LSF_REPLACE_PIM_WITH_LINUX_CGROUP parameter lets you choose which way you want to generate job process information.

Note: This parameter is only supported on Linux. The parameter is ignored on other operating systems.

The LSF cgroup feature does not support PAM jobs, so you cannot disable PIM if you run PAM jobs.

If you disable PIM with this parameter, you must use the following parameters in the `lsf.conf` file to enable cgroup memory and CPU accounting.

- **LSF_PROCESS_TRACKING=Y**
- **LSF_LINUX_CGROUP_ACCT=Y**

You can also configure the following parameters:

- **LSF_PROCESS_TRACKING=y**
- **LSB_RESOURCE_ENFORCE=memory**

The **LSB_RESOURCE_ENFORCE=memory** parameter enables memory and cgroup CPU accounting by default.

Default

N

LSF_RES_ACCT**Syntax**

LSF_RES_ACCT=*time_milliseconds* | 0

Description

If this parameter is defined, RES logs information for completed and failed tasks by default (see `lsf.acct`).

The value for LSF_RES_ACCT is specified in terms of consumed CPU time (milliseconds). Only tasks that have consumed more than the specified CPU time are logged.

If this parameter is defined as LSF_RES_ACCT=0, then all tasks are logged.

For those tasks that consume the specified amount of CPU time, RES generates a record and appends the record to the task log file `lsf.acct.host_name`. This file is located in the LSF_RES_ACCTDIR directory.

If this parameter is not defined, the LSF administrator must use the **lsadmin** command (see **lsadmin**) to turn task logging on after RES has started.

Default

Not defined

See also

LSF_RES_ACCTDIR

LSF_RES_ACCTDIR

Syntax

LSF_RES_ACCTDIR=*directory*

Description

The directory in which the RES task log file `lsf.acct.host_name` is stored.

If LSF_RES_ACCTDIR is not defined, the log file is stored in the `/tmp` directory.

Default

(UNIX)/tmp

(Windows) C:\temp

See also

LSF_RES_ACCT

LSF_RES_ACTIVE_TIME

Syntax

LSF_RES_ACTIVE_TIME=*time_seconds*

Description

Time in seconds before LIM reports that RES is down.

Minimum value

10 seconds

Default

90 seconds

LSF_RES_ALIVE_TIMEOUT

Syntax

LSF_RES_ALIVE_TIMEOUT=*time_seconds*

Description

Controls how long the task res on non-first execution hosts waits (in seconds) before cleaning up the job. If set to 0, this parameter is disabled.

Restart all res in the cluster after setting or changing this parameter.

Default

60 seconds

LSF_RES_CLIENT_TIMEOUT

Syntax

LSF_RES_CLIENT_TIMEOUT=*time_minutes*

Description

Specifies in minutes how long an application RES waits for a new task before exiting.



CAUTION:

If you use the LSF API to run remote tasks and you define this parameter with timeout, the remote execution of the new task fails (for example, `ls_rtask()`).

Default

The parameter is not set; the application RES waits indefinitely for new task to come until client tells it to quit.

LSF_RES_CONNECT_RETRY

Syntax

LSF_RES_CONNECT_RETRY=*integer* | 0

Description

The number of attempts by RES to reconnect to NIOS.

If LSF_RES_CONNECT_RETRY is not defined, the default value is used.

Default

0

See also

LSF_NIOS_RES_HEARTBEAT

LSF_RES_DEBUG

Syntax

LSF_RES_DEBUG=1 / 2

Description

Sets RES to debug mode.

If LSF_RES_DEBUG is defined, the Remote Execution Server (RES) operates in single user mode. No security checking is performed, so RES should not run as root. RES does not look in the services database for the RES service port number. Instead, it uses port number 36002 unless LSF_RES_PORT has been defined.

Specify 1 for this parameter unless you are testing RES.

Valid values

LSF_RES_DEBUG=1

RES runs in the background with no associated control terminal.

LSF_RES_DEBUG=2

RES runs in the foreground and prints error messages to `tty`.

Default

Not defined

See also

LSF_LIM_DEBUG, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_RES_PORT

See LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT.

LSF_RES_RLIMIT_UNLIM**Syntax**

LSF_RES_RLIMIT_UNLIM=cpu | fsize | data | stack | core | vmem

Description

By default, RES sets the hard limits for a remote task to be the same as the hard limits of the local process. This parameter specifies those hard limits which are to be set to unlimited, instead of inheriting those of the local process.

Valid values are `cpu`, `fsize`, `data`, `stack`, `core`, and `vmem`, for CPU, file size, data size, stack, core size, and virtual memory limits, respectively.

Example

The following example sets the CPU, core size, and stack hard limits to be unlimited for all remote tasks:

```
LSF_RES_RLIMIT_UNLIM="cpu core stack"
```

Default

Not defined

LSF_RES_STARTUP_RETRY_INTERVAL**Syntax**

LSF_RES_STARTUP_RETRY_INTERVAL=*seconds*

Description

If res failed to initialize socket, this parameter specifies the amount of time to wait before res tries again (if **LSF_RES_STARTUP_RETRY_TIMES** is specified).

Valid values

1 - INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

If **LSF_RES_STARTUP_RETRY_TIMES** is defined, the default value is 1. Otherwise, not defined.

See also

LSF_RES_STARTUP_RETRY_TIMES

LSF_RES_STARTUP_RETRY_TIMES

Syntax

LSF_RES_STARTUP_RETRY_TIMES=*integer*

Description

The maximum number of times for res to try to initialize a socket when the first attempt failed.

Valid values

0 - INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

0

res does not try to initialize a socket if the first attempt failed.

See also

LSF_RES_STARTUP_RETRY_INTERVAL

LSF_RES_SYNCUP_INTERVAL

Syntax

LSF_RES_SYNCUP_INTERVAL=*time_seconds*

Description

Time in seconds before RES contacts LIM to get the latest host configuration.

If set to 0, RES never contacts LIM to get the host configuration. This can reduce the query load on LIM, but if there are cluster-level configuration changes, you must restart RES to get the new configuration.

Minimum value

90 seconds

Default

1800 seconds (30 minutes)

LSF_RES_TIMEOUT**Syntax**

LSF_RES_TIMEOUT=*time_seconds*

Description

Timeout when communicating with RES.

Default

15

LSF_ROOT_REX**Syntax**

LSF_ROOT_REX=*all* | *text*

Description

UNIX only.

Specifies the root execution privileges for jobs from local and remote hosts.

If defined as any value in the local cluster, allows root remote execution privileges (subject to identification checking) for jobs from local hosts in the same cluster, for both interactive and batch jobs. Causes RES to accept requests from root on other local hosts in the same cluster, subject to identification checking.

If defined as *all* in IBM Spectrum LSF multicluster capability, allows root remote execution privileges (subject to identification checking) for jobs from remote and local cluster hosts, for both interactive and batch jobs. Causes RES to accept requests from the superuser (root) on local and remote hosts, subject to identification checking.

If **LSF_ROOT_REX** is not defined, remote execution requests from root are refused.

Default

Not defined. Root execution is not allowed.

Important:**See also**

LSF_AUTH, LSF_DISABLE_LSRUN

LSF_RSH**Syntax**

LSF_RSH=*command* [*command_options*]

Description

Specifies shell commands to use when the following LSF commands require remote execution:

- **badmin hstartup**

- **bpeek**
- **lsadmin limstartup**
- **lsadmin resstartup**
- **lsfrestart**
- **lsfshutdown**
- **lsfstartup**
- **lsrcp**

By default, **rsh** is used for these commands. Use LSF_RSH to enable support for **ssh**.

EGO parameter

EGO_RSH

Default

Not defined

Example

To use an **ssh** command before trying **rsh** for LSF commands, specify:

```
LSF_RSH="ssh -o 'PasswordAuthentication no' -o 'StrictHostKeyChecking no'"
```

ssh options such as **PasswordAuthentication** and **StrictHostKeyChecking** can also be configured in the global SSH_ETC/ssh_config file or \$HOME/.ssh/config.

See also

ssh, **ssh_config**

LSF_SECUREDIR

Syntax

LSF_SECUREDIR=*path*

Description

Windows only; mandatory if using `lsf.sudoers`.

Path to the directory that contains the file `lsf.sudoers` (shared on an NTFS file system).

LSF_SEND_CONFINFO_TCP_THRESHOLD

Syntax

LSF_SEND_CONFINFO_TCP_THRESHOLD=*size_in_bytes*

Description

Controls the transfer method (TCP or UDP) that the slave LIM uses to send configuration information to the master LIM.

Configuring this parameter with an integer value uses TCP to send the configuration information package if the package size is greater than this value in bytes. Otherwise, it is sent through UDP.

Valid values

Specify any integer.

Default

0. The slave LIM always uses TCP.

LSF_SERVER_HOSTS**Syntax**

```
LSF_SERVER_HOSTS="host_name ..."
```

Description

Defines one or more server hosts that the client should contact to find a Load Information Manager (LIM). LSF server hosts are hosts that run LSF daemons and provide loading-sharing services. Client hosts are hosts that only run LSF commands or applications but do not provide services to any hosts.

Important:

LSF_SERVER_HOSTS is required for non-shared slave hosts.

Use this parameter to ensure that commands execute successfully when no LIM is running on the local host, or when the local LIM has just started. The client contacts the LIM on one of the LSF_SERVER_HOSTS and execute the command, provided that at least one of the hosts defined in the list has a LIM that is up and running.

If LSF_SERVER_HOSTS is not defined, the client tries to contact the LIM on the local host.

The host names in LSF_SERVER_HOSTS must be enclosed in quotes and separated by white space. For example:

```
LSF_SERVER_HOSTS="hostA hostD hostB"
```

The parameter string can include up to 4094 characters for UNIX or 255 characters for Windows.

Interaction with LSF_MASTER_LIST

Starting in LSF 7, LSF_MASTER_LIST must be defined in `lsf.conf`. You can use the same list of hosts, or a subset of the master host list, in LSF_SERVER_HOSTS. If you include the primary master host in LSF_SERVER_HOSTS, you should define it as the last host of the list.

If LSF_ADD_CLIENTS is defined in `install.config` at installation, **lsfinstall** automatically appends the hosts in LSF_MASTER_LIST to the list of hosts in LSF_SERVER_HOSTS so that the primary master host is last. For example:

```
LSF_MASTER_LIST="lsfmaster hostE"
```

```
LSF_SERVER_HOSTS="hostB hostC hostD hostE lsfmaster"
```

```
LSF_ADD_CLIENTS="clientHostA"
```

The value of LSF_SERVER_HOSTS is not changed during upgrade.

Default

Not defined

See also**LSF_MASTER_LIST**

LSF_SERVERDIR

Syntax

LSF_SERVERDIR=*directory*

Description

Directory in which all server binaries and shell scripts are installed.

These include **lim**, **res**, **nios**, **sbatchd**, **mbatchd**, and **mbschd**. If you use **elim**, **eauth**, **eexec**, **esub**, etc, they are also installed in this directory. If you use LSF Data Manager, the transfer scripts are installed in this directory.

Default

LSF_MACHDEP/etc

See also

LSB_ECHKPNT_METHOD_DIR

LSF_SHELL_AT_USERS

Syntax

LSF_SHELL_AT_USERS="*user_name user_name ...*"

Description

Applies to **lstcsh** only. Specifies users who are allowed to use @ for host redirection. Users not specified with this parameter cannot use host redirection in **lstcsh**. To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).

If this parameter is not defined, all users are allowed to use @ for host redirection in **lstcsh**.

Default

Not defined

LSF_SHIFT_JIS_INPUT

Syntax

LSF_SHIFT_JIS_INPUT=*y | n*

Description

Enables LSF to accept Shift-JIS character encoding for job information (for example, user names, queue names, job names, job group names, project names, commands and arguments, **esub** parameters, external messages, etc.)

Default

n

LSF_STRICT_CHECKING

Syntax

LSF_STRICT_CHECKING=Y

Description

If set, enables more strict checking of communications between LSF daemons and between LSF commands and daemons when LSF is used in an untrusted environment, such as a public network like the Internet.

If you enable this parameter, you must enable it in the entire cluster, as it affects all communications within LSF. If it is used in a MultiCluster environment, it must be enabled in all clusters, or none. Ensure that all binaries and libraries are upgraded to LSF Version 7, including LSF_BINDIR, LSF_SERVERDIR and LSF_LIBDIR directories, if you enable this parameter.

If your site uses any programs that use the LSF base and batch APIs, or LSF MPI (Message Passing Interface), they need to be recompiled using the LSF Version 7 APIs before they can work properly with this option enabled.

Important:

You must shut down the entire cluster before enabling or disabling this parameter.

If LSF_STRICT_CHECKING is defined, and your cluster has slave hosts that are dynamically added, LSF_STRICT_CHECKING must be configured in the local lsf.conf on all slave hosts.

Valid value

Set to Y to enable this feature.

Default

Not defined. LSF is secure in trusted environments.

LSF_STRICT_RESREQ

When **LSF_STRICT_RESREQ=Y**, the resource requirement selection string must conform to the stricter resource requirement syntax

Syntax

LSF_STRICT_RESREQ=Y | N

Description

The strict resource requirement syntax is described in *Administering IBM Spectrum LSF*. The strict resource requirement syntax only applies to the select section. It does not apply to the other resource requirement sections (order, rusage, same, span, or cu).

When the **LSF_STRICT_RESREQ=Y** parameter is specified in the `lsf.conf` file, LSF rejects resource requirement strings with an `rusage` section that contains a non-consumable resource.

When **LSF_STRICT_RESREQ=N**, the default resource requirement selection string evaluation is performed.

Default

Set to Y at time of installation. If otherwise undefined, then N.

LSF_STRIP_DOMAIN

Syntax

```
LSF_STRIP_DOMAIN=domain_suffix[:domain_suffix ...]
```

Description

(Optional) If all of the hosts in your cluster can be reached using short host names, you can configure LSF to use the short host names by specifying the portion of the domain name to remove. If your hosts are in more than one domain or have more than one domain name, you can specify more than one domain suffix to remove, separated by a colon (:).

Example:

```
LSF_STRIP_DOMAIN=.example.com:.generic.com
```

In the above example, LSF accepts `hostA`, `hostA.example.com`, and `hostA.generic.com` as names for `hostA`, and uses the name `hostA` in all output.

The leading period '.' is required.

Setting this parameter only affects host names displayed through LSF, it does not affect DNS host lookup.

After adding or changing `LSF_STRIP_DOMAIN`, use `lsadmin reconfig` and `badmin mbdrestart` to reconfigure your cluster.

EGO parameter

EGO_STRIP_DOMAIN

Default

Not defined

LSF_TIME_CMD

Syntax

```
LSF_TIME_CMD=timing_level
```

Description

The timing level for checking how long LSF commands run. Time usage is logged in milliseconds. Specify a positive integer.

Default

Not defined

See also

`LSB_TIME_MBD`, `LSB_TIME_SBD`, `LSB_TIME_CMD`, `LSF_TIME_LIM`, `LSF_TIME_RES`

LSF_TIME_LIM

Syntax

```
LSF_TIME_LIM=timing_level
```

Description

The timing level for checking how long LIM routines run.
Time usage is logged in milliseconds. Specify a positive integer.

EGO parameter

EGO_TIME_LIM

Default

Not defined

See also

LSB_TIME_CMD, LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_RES

LSF_TIME_PIM

The timing level for checking how long PIM routines run.

Syntax

LSF_TIME_PIM=*timing_level*

Description

Time usage is logged in milliseconds. Specify a positive integer.

EGO parameter

EGO_TIME_PIM

Default

Not defined

See also

LSB_TIME_CMD, LSB_TIME_LIM, LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_RES

LSF_TIME_RES**Syntax**

LSF_TIME_RES=*timing_level*

Description

The timing level for checking how long RES routines run.
Time usage is logged in milliseconds. Specify a positive integer.
LSF_TIME_RES is not supported on Windows.

Default

Not defined

See also

LSB_TIME_CMD, LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_LIM

LSF_TMPDIR

Syntax

LSF_TMPDIR=*directory*

Description

Specifies the path and directory for temporary LSF internal files. If LSB_JOB_TMPDIR is not set, a job-specific temporary directory is also created under LSF_TMPDIR. For more details on the job-specific temporary directory, refer to **LSB_JOB_TMPDIR**.

When LSB_JOB_TMPDIR is not defined in `lsf.conf`, LSF uses the directory specified by LSF_TMPDIR (or `/tmp` by default) on execution host.

Default

By default, LSF_TMPDIR is not enabled. If LSF_TMPDIR is not specified in `lsf.conf`, this parameter is defined as follows:

- On UNIX: \$TMPDIR or `/tmp`
- On Windows: %TMP%, %TEMP%, or %SystemRoot%

LSF_UGROUP_TRANSFER

The default job execution model uses UNIX group information that is set on the execution side and cannot be set by the user. Setting the **LSF_UGROUP_TRANSFER** parameter enables users to specify a supplementary group from the client side.

Syntax

LSF_UGROUP_TRANSFER=Y|y|N|n

Description

When set to Y|y, transfers secondary user groups from the submission host to the execution host for job execution.

The order of a secondary user group is important since an NFS limitation only checks the first 16 secondary user groups for appropriate permission. Transferring the secondary group order from the submission host to the execution host addresses this limitation. However, the user must ensure the secondary user group order on the submission host is valid for the job before submission.

This does not affect primary user group behavior on the execution host.

Note: When using LSF multicluster capability, you must set this parameter to Y|y on each cluster.

Note: This feature does not work with container jobs (for example, Docker).

Default

N

LSF_UNIT_FOR_LIMITS

Enables scaling of large units in the resource usage limits.

Syntax

LSF_UNIT_FOR_LIMITS=*unit*

Description

You can scale limits for the following resource limits.

- core
- mem
- stack
- swp
- tmp

The default of KB for these limits might be too small for some environments that make use of very large resource usage limits, for example, GB or TB.

The **LSF_UNIT_FOR_LIMITS** parameter applies cluster-wide to limits at the job-level (**bsub** command), queue-level (`lsb.queues` file), and application level (`lsb.applications` file). This parameter alters the meaning of all numeric values in the `lsb.resources` file to match the specified unit in `gpool`, `limits`, and `hostexport`. It also controls the resource usage attached to the job and the memory amount that defines the size of a package in `GSLA`.

The limit unit that is specified by the **LSF_UNIT_FOR_LIMITS** parameter also applies when limits are modified with the **bmod** command. The limit unit also applies to the display of resource usage limits in query commands (**bacct**, **bapp**, **bhist**, **bhosts**, **bjobs**, **bqueues**, **lsload**, **lshosts**, and **lsmon**).

By default, the `tmp` resource is not supported by the **LSF_UNIT_FOR_LIMITS** parameter. Use the parameter **LSF_ENABLE_TMP_UNIT=Y** to enable the **LSF_UNIT_FOR_LIMITS** parameter to support limits on the `tmp` resource.

Important: Before you change the units of your resource usage limits, completely drain the cluster of all workload so that no running, pending, or finished jobs are in the system.

In the LSF multicluster capability environment, configure the same unit for all clusters.

Note: Other limits (such as the file limit) are not affected by setting the parameter **LSF_UNIT_FOR_LIMITS**.

Example

A job is submitted with **bsub -M 100** and the **LSF_UNIT_FOR_LIMITS=GB** parameter is set. The memory limit for the job is 100 GB rather than the default 100 MB.

Valid values

The *unit* value indicates the unit for the resource usage limit, one of:

- KB or K (kilobytes)
- MB or M (megabytes)
- GB or G (gigabytes)
- TB or T (terabytes)
- PB or P (petabytes)
- EB or E (exabytes)
- ZB or Z (zettabytes)

Default

Set to MB at time of installation. If the **LSF_UNIT_FOR_LIMITS** parameter is not defined in the `lsf.conf` file, the default settings are as follows:

- **bsub** and **bmod**
 - **bsub -C**: KB
 - **bmod -C** : KB

lsf.conf

- **core_limit -M:** KB
- **mem_limit -S:** KB
- **stack_limit -v:** KB
- **swap_limit:** KB
- **bsub -R:** MB
 - mem, swp and tmp in **RUSAGE** and select sections: MB
- **lsb.applications**
 - **CORELIMIT, MEMLIMIT, STACKLIMIT, SWAPLIMIT:** KB
 - **RES_REQ**
 - mem, swp, and tmp in **RUSAGE** and select sections: MB
- **lsb.queues**
 - **CORELIMIT, MEMLIMIT, STACKLIMIT, SWAPLIMIT:** KB
 - **RES_REQ**
 - mem, swp, and tmp in **RUSAGE** and select sections: MB
 - mem, swp, and tmp in **RESRSV_LIMIT:** MB
 - MEM and SWP that is configured in **RSRC**, found in **APS_PRIORITY:** MB
- **lsb.resources**
 - MEM, SWP and TMP in Limit section: MB
 - mem for package resource TYPE in Guaranteed section: MB
 - MEM and SWP in the Host Export section: MB

LSF_UDP_PORT_RANGE

Syntax

LSF_UDP_PORT_RANGE="*min_port_number-max_port_number*"

Description

UDP port range to be used by the LSF daemons. The UDP socket for the LSF daemons binds one port in the specified range.

For the minimum port number (*min_port_number*), specify an integer in the range 1024 - 65526 . For the maximum port number (*max_port_number*), specify an integer in the range 1033 - 65535. Include at least 10 ports in this range.

If not defined, LSF uses a random port in the range 1024 - 65535.

Default

Not defined.

LSF_USE_HOSTEQUIV

Syntax

LSF_USE_HOSTEQUIV=y | Y

Description

(UNIX only; optional)

If `LSF_USE_HOSTEQUIV` is defined, `RES` and `mbatchd` call the `ruserok()` function to decide if a user is allowed to run remote jobs.

The `ruserok()` function checks in the `/etc/hosts.equiv` file and the user's `$HOME/.rhosts` file to decide if the user has permission to execute remote jobs.

If `LSF_USE_HOSTEQUIV` is not defined, all normal users in the cluster can execute remote jobs on any host.

If `LSF_ROOT_REX` is set, `root` can also execute remote jobs with the same permission test as for normal users.

Default

Not defined

See also

LSF_ROOT_REX

LSF_USER_DOMAIN

Syntax

`LSF_USER_DOMAIN=domain_name:domain_name:domain_name... .`

Description

Enables the UNIX/Windows user account mapping feature, which allows cross-platform job submission and execution in a mixed UNIX/Windows environment. `LSF_USER_DOMAIN` specifies one or more Windows domains that LSF either strips from the user account name when a job runs on a UNIX host, or adds to the user account name when a job runs on a Windows host.

Important:

Configure **LSF_USER_DOMAIN** immediately after you install LSF; changing this parameter in an existing cluster requires that you verify and possibly reconfigure service accounts, user group memberships, and user passwords.

Specify one or more Windows domains, separated by a colon (:). You can enter an unlimited number of Windows domains. A period (.) specifies a local account, not a domain.

Examples

`LSF_USER_DOMAIN=BUSINESS`

`LSF_USER_DOMAIN=BUSINESS:ENGINEERING:SUPPORT`

Default

The default depends on your LSF installation:

- If you upgrade a cluster to LSF version 7, the default is the existing value of **LSF_USER_DOMAIN**, if defined
- For a new cluster, this parameter is not defined, and UNIX/Windows user account mapping is not enabled

LSF_VPLUGIN

The full path to the vendor MPI library `libxmpi.so`.

Syntax

`LSF_VPLUGIN=path`

Description

This path is used with LSF HPC features.

Examples

- IBM Spectrum LSF MPI: `LSF_VPLUGIN=/opt/mpi/lib/pa1.1/libmpirm.sl`
- MPI: `LSF_VPLUGIN=/usr/lib32/libxmpi.so`
- Linux (64-bit x-86 Linux 2.6, glibc 2.3.): `LSF_VPLUGIN=/usr/lib32/libxmpi.so:/usr/lib/libxmpi.so:/usr/lib64/libxmpi.so`

Default

Not defined

LSF_WINDOWS_HOST_TYPES**Syntax**

```
LSF_WINDOWS_HOST_TYPES="HostType1 HostType2 HostType3 ..."
```

Description

Use this parameter to set the Windows host type in mixed cluster environments, with a UNIX master host and Windows clients. Set **LSF_WINDOWS_HOST_TYPES** in `lsf.conf` to configure Windows host types.

If not defined (and Windows clients are other than the default (NTX86, NTX64, and NTIA64), running **lspasswd** on the Windows server returns an error message.

Except for "NTX86", "NTX64", and "NTIA64", all Windows host types defined in **LSF_WINDOWS_HOST_TYPES** must be defined in the HostType section in `lsf.shared` file. If not defined in HostType, LSF issues a warning message (except for the three default types) when starting or restarting `lim`.

After changing **LSF_WINDOWS_HOST_TYPES**, run **lsadmin limrestart** for changes to take effect.

Examples

If **LSF_WINDOWS_HOST_TYPES**="NTX86 NTX64 aaa" but "aaa" is not defined in `lsf.shared`. After `lim` startup, the log file will show:

```
Feb 27 05:09:10 2013 15150 3 1.2.7 dotypelist(): /.../conf/lsf.shared: The host
type defined by LSF_WINDOWS_HOST_TYPES aaa should also be defined in
lsf.shared.
```

Default

```
LSF_WINDOWS_HOST_TYPES="NTX86 NTX64 NTIA64"
```

MQTT_BROKER_HOST

For LSF resource connector. If you do not use the MQTT message broker daemon (**mosquitto**) that is provided with LSF, specifies the host name that **mosquitto** runs on. The MQTT message broker receives provider host information from **ebrokerd** and publishes that information for the **bhosts -rc** and **bhosts -rconly** command options to display.

Syntax

```
MQTT_BROKER_HOST=host_name
```

Description

If you use an existing MQTT message broker, use the **MQTT_BROKER_HOST** parameter to specify the host that runs the **mosquitto** daemon. You can also optionally specify a port for the MQTT broker with the **MQTT_BROKER_PORT** parameter.

After configuring this parameter, run **badadmin mbdrestart** on the master host to restart **ebrokerd**.

Example

```
MQTT_BROKER_HOST=hosta
```

Default

Not defined.

See also

- **EBROKERD_HOST_CLEAN_DELAY**
- **LSF_MQ_BROKER_HOSTS**
- **MQTT_BROKER_PORT**
- **bhosts -rc** and **bhosts -rconly**

MQTT_BROKER_PORT

For LSF resource connector. If you do not use the MQTT message broker daemon (**mosquitto**) that is provided with LSF, specifies the TCP port for the MQTT message broker daemon (**mosquitto**). The MQTT message broker receives provider host information from **ebrokerd** and publishes that information for the **bhosts -rc** and **bhosts -rconly** command options to display.

Syntax

```
MQTT_BROKER_PORT=port_number
```

Description

After configuring this parameter, run **badadmin mbdrestart** on the master host to restart **ebrokerd**.

If you use the **mosquitto** daemon from the LSF distribution, and you define a port that is not the default 1883, you also must define the port in `$LSF_ENVDIR/mosquitto.conf`. LIM then starts the **mosquitto** daemon with the non-default port and **ebrokerd** connects the MQTT broker via the port defined in the `lsf.conf` file.

If you are using an external **mosquitto** daemon, the port number you define in **MQTT_BROKER_PORT** must be the same port in the external `mosquitto.conf` file.

Example

```
MQTT_BROKER_PORT=4477
```

Default

1883.

See also

- **EBROKERD_HOST_CLEAN_DELAY**
- **LSF_MQ_BROKER_HOSTS**
- **MQTT_BROKER_HOST**

- **bhosts -rc** and **bhosts -rconly**

RC_MAX_AUDIT_LOG_KEEP_TIME

Syntax

RC_MAX_AUDIT_LOG_KEEP_TIME=*integer*

Description

Specifies the amount of time that LSF keeps the resource connector audit log files, in months.

Default

Not defined.

See also

LSF_RC_AUDIT_LOG, RC_MAX_AUDIT_LOG_SIZE

RC_MAX_AUDIT_LOG_SIZE

Syntax

RC_MAX_AUDIT_LOG_SIZE=*integer*

Description

Specifies the maximum size of the LSF resource connector audit log files, in MB.

Default

Not defined.

See also

LSF_RC_AUDIT_LOG, RC_MAX_AUDIT_LOG_KEEP_TIME

XLSF_APPDIR

Syntax

XLSF_APPDIR=*directory*

Description

(UNIX only; optional) Directory in which X application default files for LSF products are installed.

The LSF commands that use X look in this directory to find the application defaults. Users do not need to set environment variables to use the LSF X applications. The application default files are platform-independent.

Default

LSF_INDEP/misc

XLSF_UIDDIR

Syntax

XLSF_UIDDIR=*directory*

Description

(UNIX only) Directory in which Motif User Interface Definition files are stored. These files are platform-specific.

Default

LSF_LIBDIR/uid

MC_PLUGIN_REMOTE_RESOURCE**Syntax**

MC_PLUGIN_REMOTE_RESOURCE=y

Description

MultiCluster job forwarding model only. By default, the submission cluster does not consider remote resources. Define MC_PLUGIN_REMOTE_RESOURCE=y in the submission cluster to allow consideration of remote resources.

Note:

When MC_PLUGIN_REMOTE_RESOURCE is defined, only the following resource requirements (boolean only) are supported: -R "type==*type_name*", -R "same[*type*]" and -R "defined(*resource_name*)"

Note:

When **MC_PLUGIN_SCHEDULE_ENHANCE** in `lsb.params` is defined, remote resources are considered as if MC_PLUGIN_REMOTE_RESOURCE=Y regardless of the actual value. In addition, details of the remote cluster workload are considered by the submission cluster scheduler.

Default

Not defined. The submission cluster does not consider remote resources.

See also

MC_PLUGIN_SCHEDULE_ENHANCE in `lsb.params`

lsf.datamanager

The `lsf.datamanager` file controls the operation of IBM Spectrum LSF Data Manager features. Each cluster has one LSF data management configuration file, called `lsf.datamanager.cluster_name`. The *cluster_name* suffix is the name of the cluster that is defined in the `Cluster` section of the `lsf.shared` file. The file is read by the LSF data management daemon **dmd**. Since one LSF data manager can serve multiple LSF clusters, the contents of this file must be identical on each cluster that shares LSF data manager.

Changing lsf.datamanager configuration

After you change `lsf.datamanager.cluster_name`, run the **bdata admin reconfig** command to reconfigure LSF data manager on all candidate hosts.

Location

The `lsf.datamanager` file is located in the directory that is defined by the **LSF_ENVDIR** parameter.

Structure

The `lsf.datamanager.cluster_name` file contains two configuration sections:

Parameters section

The Parameters section of the `lsf.datamanager` file configures LSF data manager operations.

RemoteDataManagers section

Optional. The RemoteDataManagers section tells a local LSF data manager how to communicate with remote LSF data managers in LSF multicluster capability forwarding clusters. Only the cluster that is sending jobs needs to configure the RemoteDataManagers section.

Parameters section

The Parameters section of the `lsf.datamanager` file configures LSF data manager operations.

Description

The Parameters section of the `lsf.datamanager` file configures the following operations:

- LSF data manager administrators
- The default file transfer command
- The location of the LSF data manager staging area
- File permissions on the LSF data manager cache
- The grace period for LSF data manager cache cleanup

Parameters

- **ADMINS**
- **CACHE_ACCESSIBLE_FILES**
- **CACHE_ACCESS_CONTROL**
- **CACHE_INPUT_GRACE_PERIOD**
- **CACHE_OUTPUT_GRACE_PERIOD**
- **CACHE_PERMISSIONS**
- **CACHE_REFRESH_INTERVAL**
- **FILE_TRANSFER_CMD**
- **PERMISSION_MASK**
- **QUERY_NTHREADS**
- **REMOTE_CACHE_REFRESH_INTERVAL**
- **STAGING_AREA**

ADMINS

Required. List of LSF data manager administrator users.

Syntax

`ADMINS=user_name [user_name ...]`

Description

The specified users can reconfigure and shut down the LSF data manager daemon (**dmd**), and use the **bdata tags** subcommand to list or clean intermediate files that are associated with a tag for users. LSF data manager administrators must be administrators in each of the connected LSF clusters.

You cannot configure the root user as an LSF data manager administrator, but root has the same privileges as an LSF data manager administrator.

The parameter takes effect after you restart or reconfigure LSF data manager.

Valid values

Any LSF administrator user account, except root.

Example

```
ADMINS=lsfadmin user1 user2
```

Default

None. Required parameter.

CACHE_ACCESS_CONTROL

Enables the owner of a data tag or a data file in the cache to change the user group ownership of a tag, an ownership, and the permissions of a file.

Syntax

```
CACHE_ACCESS_CONTROL=Y | y | N | n
```

Description

LSF data manager cache access control allows a user to access cached data files that belong to a user in another user group.

The **CACHE_PERMISSIONS=a11** parameter and the **CACHE_ACCESSIBLE_FILES= Y** parameter must be configured to enable cache access control.

Default

N - changing the user group of a cached file or tag is not enabled.

CACHE_ACCESSIBLE_FILES

Specifies whether LSF data manager initiates a transfer job or creates a symbolic link from the cache to the requested data file.

Syntax

```
CACHE_ACCESSIBLE_FILES=Y | y | N | n
```

Description

By default, if LSF data manager can directly access a required data file through the file system, the **CACHE_ACCESSIBLE_FILES=N** parameter avoids unnecessary local stage in of the file. Creating a symbolic link to a locally accessible file increases job throughput because the job does not need to wait for a transfer job to finish. When the **CACHE_ACCESSIBLE_FILES=Y** parameter is configured, a transfer job is always initiated for jobs that are submitted with data requirements, and automatic symbolic linking for accessible files is disabled. Always starting a transfer job might be desirable to ensure that the latest copy of the data file is made available to the job.

Default

N - a symbolic link from the cache is created for files that LSF data manager can access directly.

CACHE_INPUT_GRACE_PERIOD

Minimum time in minutes that an input file is kept in the LSF data manager cache after no jobs reference it.

Syntax

```
CACHE_INPUT_GRACE_PERIOD=minutes
```

Description

After the specified number of minutes, the job that is associated with the input file can no longer be queried through the **bdata** command. The files that the job requested to be staged in are physically deleted from the cache.

The grace period for the input file begins when no data jobs request the file and no transfer jobs are transferring it. The input file grace period does not apply to transfers with ERROR status.

The parameter takes effect after you restart or reconfigure LSF data manager.

Valid values

1 - 2147483647 minutes

Default

1440 minutes (1 day)

CACHE_OUTPUT_GRACE_PERIOD

Minimum time in minutes that an output file is kept in the LSF data manager cache after its transfer completes, either successfully or unsuccessfully.

Syntax

CACHE_OUTPUT_GRACE_PERIOD=*minutes*

Description

After the specified number of minutes, the job that is associated with the output file can no longer be queried through the **bdata** command. The files that the job requested to be staged out are physically deleted from the cache.

The grace period for the output file begins when all the output file records associated to the same job reach TRANSFERRED or ERROR status. However, the files and job records are not cleaned up until the grace periods expire for all stage-out requirements that are associated with the same job. Output files can be queried until the grace period expires for *all* output file records associated with the job.

The grace period on the output file begins whether the file transfer completes successfully or unsuccessfully. Files are cleaned up only after the grace periods of all stage-out requirements for the job expire. After the grace period expires, the stage-out records are removed, and you cannot search the stage-out history from LSF data manager.

The grace period does not apply to files uploaded to the cache with the **bstage out -tag** command. You must use the **bdata tags clean** command to clean up these files manually.

The parameter takes effect after you restart or reconfigure LSF data manager.

Valid values

1 - 2147483647 minutes

Default

180 minutes (3 hours)

CACHE_PERMISSIONS

Sets file permissions and ownership of the LSF data manager staging area subdirectories.

Syntax

CACHE_PERMISSIONS=*user | group | all*

Description

CACHE_PERMISSIONS=user

By default, files are stored in a user-specific subdirectory in the staging area. All files are owned by that user with 700 file permission mode. If two users ask for the same file to be pre-staged with the **bsub -data** command, LSF transfers a separate copy for each user. Different users cannot use the same cached file or query the staged files for other users with the **bdata cache** command.

CACHE_PERMISSIONS=group

Users in the same primary UNIX user group can share the cached files to avoid unnecessary file transfers. Files are owned by the first user in the group that requests the file. The subdirectory is based on the main group of the transferring user, and file permissions are set to 750. Users in the same primary group can query files for their group with the **bdata cache** command.

When **CACHE_PERMISSIONS=group** is specified, the **-g** option of the **bdata** command shows the cached files that belong to the specified user group.

CACHE_PERMISSIONS=all

Only a single cache is created for incoming files, and files are shareable by all users. All files pre-staged with the **bsub -data** command are stored in the staging area with permission 755. Files are owned by the first user who requests the file. If two users ask for the same file to be pre-staged, only one copy of the file is pre-staged, owned by the first requesting user. Users can query any file in the cache with the **bdata cache** command.

Note: The value of **CACHE_PERMISSIONS** affects the directory structure of the staging area cache, which LSF data manager depends on for recovery. To avoid losing the cache, do not change the value of the **CACHE_PERMISSIONS** parameter between LSF data manager restarts.

If you must change the value of the **CACHE_PERMISSIONS** parameter after the cache is already in use, files in the cache subdirectory `$STAGING_AREA/stgin/$CACHE_PERMISSIONS` corresponding to the old value are not cleaned up by LSF data manager. You must manually delete them as root. Be careful that no running jobs are using them.

The parameter takes effect after you restart or reconfigure LSF data manager.

Default

user

CACHE_REFRESH_INTERVAL

Limits the number of transfer jobs to the data manager by setting a refresh interval for the file cache.

Syntax

`CACHE_REFRESH_INTERVAL=minutes`

Description

Every time a file or folder is requested by a new job, there could be a new transfer job. The **CACHE_REFRESH_INTERVAL** parameter limits the number of times that LSF Data Manager performs data copy operations to the cache area. If a data job requests source data that is already in the cache and is within the time period as specified in **CACHE_REFRESH_INTERVAL**, the transfer job does not copy the data to the staging area to refresh the cache. In addition, the transfer job does not refresh a new copy of the source data in the cache if the source file is not changed.

If **CACHE_ACCESS_CONTROL=Y** is configured, the data manager ignores the **CACHE_REFRESH_INTERVAL** parameter value and copies the source data file to cache if the source data file is changed, otherwise the transfer job does not copy the data. The data manager also ignores the **CACHE_REFRESH_INTERVAL** parameter value if **LSF_DATA_BSUB_CHKSUM** is defined as **y** or **Y**.

Valid values

0 - 2147483647 (minutes)

Default

15

FILE_TRANSFER_CMD

The command that is used by LSF data manager to transfer data files.

Syntax

FILE_TRANSFER_CMD=*command*

Description

The specified command must take two arguments of the form [*host_name:*]*abs_file_path*. The first argument is an absolute path to the location of the source file and the second is absolute path to the destination of the transfer.

The command must be able to accept path descriptors with or without host names for each of its two arguments. For example, the default **scp** command satisfies both requirements. The **cp** command is not valid because it can't accept a host name.

The command that you specify must block until the transfer is successfully completed or an error occurs. It must return 0 if successful and a nonzero value if an error occurs. Provide a full path to the command so that it can be accessed from the hosts that the data transfer queue points to.

If the command returns successfully, LSF data manager assumes that the transfer was completed without error.

The parameter takes effect after you restart or reconfigure LSF data manager.

Use the **SNDJOBS_TO** to enable an LSF data manager data transfer queue as a remote send-jobs queue in the execution cluster.

If you specify a remote queue with the **SNDJOBS_TO** parameter in the data transfer queue in the execution cluster, the path of the **FILE_TRANSFER_CMD** parameter must exist in the submission cluster.

Note: If you change the **FILE_TRANSFER_CMD** parameter, transfer jobs that are submitted before the parameter changed continue to use the old value. You must kill these jobs and any dependent data jobs, and submit new data jobs for the files to be transferred with the new command.

Default

/usr/bin/scp

PERMISSION_MASK

Controls how data cache file permissions can be changed by the **bdata chmod** command.

Syntax

PERMISSION_MASK=*octal_mode*

Description

The *octal-mode* is three octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. The first digit selects permissions for the user who owns the file: read (4), write (2), and execute (1). The second digit selects the permissions for other users in the group that is associated with the file. The third digit selects the permissions for other users not in the group that is associated with the file.

For example, PERMISSION_MASK=750 means that file owners can modify the rwx bits for the user, the rw bits for the group, but not the global bits. The PERMISSION_MASK=777 parameter value means that file owners can modify all the permission bits. The PERMISSION_MASK=000 value means that file owners cannot modify permissions at all.

The permissions on the directory that the file is in are changed to be consistent with the file permission, but the owner always has full access at the directory level. If the permissions on the file allow the group

or others permission to read, write, or execute, the same permissions are enabled on the directory. For example, if the permission mask is set to `PERMISSION_MASK=777` and the file mode is `570` (`rxwx---`), the permissions on the directory are changed to `770` (`rxwxr-x---`).

The parameter takes effect after you restart or reconfigure LSF data manager.

Default

When the **CACHE_ACCESS_CONTROL=Y** parameter is set, the default permission mask value is `777`. When the **CACHE_ACCESS_CONTROL** parameter is not set to `Y`, the permission mask is set to `000` regardless of configuration.

QUERY_NTHREADS

Number of threads in the LSF data manager client query thread pool.

Syntax

`QUERY_NTHREADS=integer`

Description

Increase the value of the **QUERY_NTHREADS** parameter to improve the responsiveness of the local LSF data manager to requests by remote data managers and **bdata** command clients.

The parameter takes effect after you restart or reconfigure LSF data manager.

Valid values

1 - 2147483647

Default

4

REMOTE_CACHE_REFRESH_INTERVAL

Number of seconds that information about remote LSF data manager file availability is considered to be fresh.

Syntax

`REMOTE_CACHE_REFRESH_INTERVAL=seconds`

Description

The **REMOTE_CACHE_REFRESH_INTERVAL** parameter affects how often LSF data manager queries LSF data managers in other clusters. After the specified number of seconds elapses, the job assumes that the remote cluster information about this file is stale, and queries the cluster for availability again.

The parameter takes effect after you restart or reconfigure LSF data manager.

Valid values

1 - 2147483647 seconds

Default

15 seconds

STAGING_AREA

Absolute path to the top of the data management staging area as it is accessed from the LSF data manager hosts.

Syntax

```
STAGING_AREA=[host_name:]abs_file_path
```

Description

Any host that the LSF data manager runs on must statically mount a staging area. This path must not point to a directory that has the sticky bit set in any directory in its path (such as the /tmp directory or any subdirectory of /tmp).

Check the entries in the /etc/mstab file and see whether the mounted file system source location is a prefix of host_name:path. If so, replace this prefix with the local mount point for the source location that is specified in the /etc/mstab file and access the file at the resulting path.

The resolved directory must exist.

You must restart the LSF data manager daemon (**dmd**) for this parameter to take effect.

Note: If you change the **STAGING_AREA** parameter, files that are stored in the previous cache are not recovered, and therefore are never deleted by the LSF data manager.

Data transfer hosts

Each cluster that uses LSF data management must have a set of hosts that act as data transfer nodes (also referred to as *I/O nodes*). These hosts must have the following properties:

- Each node must have read/write access to the mounted staging area.
- For input data requirements (data stage in), data transfer nodes must be able to reach the data source location through the configured transfer tool.
- For output data requirements (data stage out), data transfer nodes must be able to reach the data destination location through the configured transfer tool.
- These nodes must statically mount the staging area at the same location as the LSF data manager host.
- These nodes must be configured as LSF execution hosts in the data transfer queue.

Data transfer nodes are execution hosts for transfer job that LSF data manager submits to a data transfer queue that specifies the **DATA_TRANSFER=Y** parameter. Specify the LSF server hosts in the **HOSTS** parameter in the `lsb.queues` file for the data transfer queue.

Example

For the following staging area configuration

```
STAGING_AREA=hostA.company.com:/vol/dmd_cache
```

The following /etc/mstab file entries are valid:

- `hostA.company.com:/vol/ /mnt/vol1 0 0` - The staging area is accessed locally at `/mnt/vol1/dmd_cache`
- `hostA:/vol/dmd_cache /mnt/staging 0 0` - The staging area is accessed locally at `/mnt/staging`

The following /etc/mstab file entry is not valid:

```
hostA.company.com:/vol/vol1 /mnt/staging 0 0
```

In this case, the export location is not an ancestor of the location pointed to by the **STAGING_AREA** parameter.

Default

None. Required parameter.

RemoteDataManagers section

Optional. The `RemoteDataManagers` section of the `lsf.datamanager` file tells a local LSF data manager how to communicate with remote LSF data managers in LSF multicluster capability forwarding clusters. Only the cluster that is sending jobs needs to configure the `RemoteDataManagers` section.

Before a job with data requirements is forwarded to another cluster, the local LSF data manager contacts remote LSF data managers, which serve as candidate clusters for job forwarding. The local `dmd` daemon collects the information that is needed for the scheduler forwarding decision. If the `dmd` daemon of a candidate forwarding cluster is not configured in the section, LSF excludes that cluster from being scheduled for job forwarding.

Configure `RemoteDataManagers` section only in LSF multicluster capability submission clusters. When the `RemoteDataManagers` section is configured, LSF maintains a connection between the master LSF data manager in each cluster, and uses this connection to query the availability of remote files.

By default, the local cluster can obtain information about all other clusters that are specified in the `lsf.shared` file. The `RemoteDataManagers` section limits the clusters that the local cluster can obtain information about.

Every cluster in the `lsf.shared` file that has LSF data management features enabled must appear in the list of clusters.

You can configure only one `RemoteDataManagers` section in the `lsf.datamanager` file. If more than one `RemoteDataManagers` section exists, only the first section takes effect. Sections beyond the first one are ignored. Duplicate cluster names are ignored, and the first cluster name in the list is used.

The `bdata connections` command reports the connection status of each remote LSF data manager. If the configuration has errors, the corresponding manager shows as disconnected (`disc`) in the output of the `bdata connections` command.

The first line consists of the following required keywords:

- **CLUSTERNAME**
- **SERVERS**
- **PORT**

Subsequent lines specify the cluster names, server hosts, and ports for the remote LSF data managers.

Example RemoteDataManagers section format

```
Begin RemoteDataManagers
CLUSTERNAME  SERVERS                PORT
cluster1     (host11 host12 ... host1n)    1729
cluster2     (host21 host22 ... host2n)    4104
clusterM     (hostM1 hostM2 ... hostMn)    13832
End RemoteDataManagers
```

RemoteDataManagers parameters**CLUSTERNAME**

Specify a list of the clusters that you want the local cluster to recognize as LSF data managers.

SERVERS

The list of servers must correspond to the value of the `LSF_DATA_HOSTS` parameter in the `lsf.conf` file of the appropriate cluster.

PORT

The port number for the cluster must correspond to the value of the `LSF_DATA_PORT` parameter in the `lsf.conf` file of the appropriate cluster.

lsf.licensescheduler

The `lsf.licensescheduler` file contains LSF License Scheduler configuration information. All sections except ProjectGroup are required. In cluster mode, the Project section is also not required.

Changing lsf.licensescheduler configuration

After making any changes to `lsf.licensescheduler`, run the following commands:

- If you added, changed, or deleted the **FEATURE_DELTA** parameter, reconfigure the **bld** daemon.
- If you have added, changed, or deleted any Feature or Projects sections, you may need to restart the **mbatchd** daemon. In this case a message is written to the log file prompting the restart.

To restart the **mbatchd** daemon on each LSF cluster, run `badmin mbdrestart`.

Parameters section

Description

Required. Defines License Scheduler configuration parameters. If there is a parameter with the same name in the Feature section, setting these parameters in the Feature section overrides the global setting in the Parameters section.

Parameters section structure

The **Parameters** section begins and ends with the lines `Begin Parameters` and `End Parameters`. Each subsequent line describes one configuration parameter. The following parameters are required:

```
Begin Parameters
ADMIN=lsadmin
HOSTS=hostA hostB hostC
LMSTAT_PATH=/etc/flexlm/bin
RLMSTAT_PATH=/etc/rlm/bin
LM_STAT_INTERVAL=30
PORT=9581
End Parameters
```

Parameters

- ADMIN
- AUTH
- BLC_HEARTBEAT_FACTOR
- CHECKOUT_FROM_FIRST_HOST_ONLY
- CLUSTER_MODE
- DEMAND_LIMIT
- DISTRIBUTION_POLICY_VIOLATION_ACTION
- ENABLE_INTERACTIVE
- FAST_DISPATCH
- HEARTBEAT_INTERVAL
- HEARTBEAT_TIMEOUT
- HIST_HOURS
- HOSTS
- INUSE_FROM_RUSAGE
- LIB_CONNTIMEOUT
- LIB_RECVTIMEOUT

- LM_REMOVE_INTERVAL
- LM_REMOVE_SUSP_JOBS
- LM_REMOVE_SUSP_JOBS_INTERVAL
- LM_RESERVATION
- LM_STAT_INTERVAL
- LM_STAT_TIMEOUT
- LM_TYPE
- LMREMOVE_SUSP_JOBS
- LMREMOVE_SUSP_JOBS_INTERVAL
- LMSTAT_PATH
- LOG_EVENT
- LOG_INTERVAL
- LS_DEBUG_BLC
- LS_DEBUG_BLD
- LS_ENABLE_MAX_PREEMPT
- LS_LOG_MASK
- LS_MAX_STREAM_FILE_NUMBER
- LS_MAX_STREAM_SIZE
- LS_MAX_TASKMAN_PREEMPT
- LS_MAX_TASKMAN_SESSIONS
- LS_STREAM_FILE
- LS_PREEMPT_PEER
- MBD_HEARTBEAT_INTERVAL
- MBD_REFRESH_INTERVAL
- MERGE_BY_SERVICE_DOMAIN
- PEAK_INUSE_PERIOD
- PORT
- PREEMPT_ACTION
- PROJECT_GROUP_PATH
- REMOTE_LMSTAT_PROTOCOL
- RLMSTAT_PATH
- STANDBY_CONNTIMEOUT

ADMIN

Syntax

ADMIN=*user_name* ...

Description

Defines the LSF License Scheduler administrator using a valid UNIX user account. You can specify multiple accounts.

Used for both project mode and cluster mode.

AUTH

Syntax

AUTH=Y | N

Description

Enables LSF License Scheduler user authentication for projects for **taskman** jobs.

Used for both project mode and cluster mode.

BLC_HEARTBEAT_FACTOR

Syntax

BLC_HEARTBEAT_FACTOR=*integer*

Description

Enables **bld** to detect **blcollect** failure. Defines the number of times that **bld** receives no response from a license collector daemon (**blcollect**) before **bld** resets the values for that collector to zero. Each license usage reported to **bld** by the collector is treated as a heartbeat.

Used for both project mode and cluster mode.

Default

3

CHECKOUT_FROM_FIRST_HOST_ONLY

Syntax

CHECKOUT_FROM_FIRST_HOST_ONLY=Y | N

Description

If enabled (CHECKOUT_FROM_FIRST_HOST_ONLY=Y), LSF License Scheduler only considers user@host information for the first execution host of a parallel job when merging the license usage data. Setting in individual Feature sections overrides the global setting in the Parameters section.

If disabled, LSF License Scheduler attempts to check out user@host keys in the parallel job constructed using the user name and all execution host names, and merges the corresponding checkout information on the service domain if found. In addition, if MERGE_BY_SERVICE_DOMAIN=Y is defined, LSF License Scheduler merges multiple user@host data for parallel jobs across different service domains.

Default

Undefined (N). LSF License Scheduler attempts to check out user@host keys in the parallel job constructed using the user name and all execution host names, and merges the corresponding checkout information on the service domain if found.

CLUSTER_MODE

Syntax

CLUSTER_MODE=Y | N

Description

Enables cluster mode (instead of project mode) in License Scheduler. Setting in individual Feature sections overrides the global setting in the Parameters section.

Cluster mode emphasizes high utilization of license tokens above other considerations such as ownership. License ownership and sharing can still be configured, but within each cluster instead of across multiple clusters. Preemption of jobs (and licenses) also occurs within each cluster instead of across clusters.

Default

Not defined (N). LSF License Scheduler runs in project mode.

DEMAND_LIMIT**Syntax**

DEMAND_LIMIT=*integer*

Description

Sets a limit to which LSF License Scheduler considers the demand by each project in each cluster when allocating licenses.

Used for fast dispatch project mode only.

When enabled, the demand limit helps prevent LSF License Scheduler from allocating more licenses to a project than can actually be used, which reduces license waste by limiting the demand that LSF License Scheduler considers. This is useful in cases when other resource limits are reached, LSF License Scheduler allocates more tokens than IBM Spectrum LSF can actually use because jobs are still pending due to lack of other resources.

When disabled (that is, DEMAND_LIMIT=0 is set), LSF License Scheduler takes into account all the demand reported by each cluster when scheduling.

DEMAND_LIMIT does not affect the DEMAND that **blstat** displays. Instead, **blstat** displays the entire demand sent for a project from all clusters. For example, one cluster reports a demand of 15 for a project. Another cluster reports a demand of 20 for the same project. When LSF License Scheduler allocates licenses, it takes into account a demand of five from each cluster for the project and the DEMAND that **blstat** displays is 35.

Periodically, each cluster sends a demand for each project. This is calculated in a cluster for a project by summing up the rusage of all jobs of the project pending due to lack of licenses. Whether to count a job's rusage in the demand depends on the job's pending reason. In general, the demand reported by a cluster only represents a potential demand from the project. It does not take into account other resources that are required to start a job. For example, a demand for 100 licenses is reported for a project. However, if LSF License Scheduler allocates 100 licenses to the project, the project does not necessarily use all 100 licenses due to slot available, limits, or other scheduling constraints.

In project mode and fast dispatch project mode, **mbatchd** in each cluster sends a demand for licenses from each project. In project mode, LSF License Scheduler assumes that each project can actually use the demand that is sent to it. In fast dispatch project mode, **DEMAND_LIMIT** limits the amount of demand from each project in each cluster that is considered when scheduling.

Default

5

DISTRIBUTION_POLICY_VIOLATION_ACTION**Syntax**

DISTRIBUTION_POLICY_VIOLATION_ACTION=(PERIOD *reporting_period* CMD *reporting_command*)

reporting_period

Specify the keyword PERIOD with a positive integer representing the interval (a multiple of **LM_STAT_INTERVAL** periods) at which LSF License Scheduler checks for distribution policy violations.

reporting_command

Specify the keyword CMD with the directory path and command that LSF License Scheduler runs when reporting a violation.

Description

Defines how LSF License Scheduler handles distribution policy violations. Distribution policy violations are caused by non-LSF workloads; License Scheduler explicitly follows its distribution policies.

LSF License Scheduler reports a distribution policy violation when the total number of licenses given to the LSF workload, both free and in use, is less than the LSF workload distribution specified in **WORKLOAD_DISTRIBUTION**. If LSF License Scheduler finds a distribution policy violation, it creates or overwrites the `LSF_LOGDIR/bld.violation.service_domain_name.log` file and runs the user command specified by the **CMD** keyword.

Used for project mode only.

Example

The `LicenseServer1` service domain has a total of 80 licenses, and its workload distribution and enforcement is configured as follows:

```
Begin Parameter
...
DISTRIBUTION_POLICY_VIOLATION_ACTION=(PERIOD 5 CMD /bin/mycmd)
...
End Parameter
```

```
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1(Lp1 1 Lp2 2)
WORKLOAD_DISTRIBUTION=LicenseServer1(LSF 8 NON_LSF 2)
End Feature
```

According to this configuration, 80% of the available licenses, or 64 licenses, are available to the LSF workload. LSF License Scheduler checks the service domain for a violation every five scheduling cycles, and runs the `/bin/mycmd` command if it finds a violation.

If the current LSF workload license usage is 50 and the number of free licenses is 10, the total number of licenses assigned to the LSF workload is 60. This is a violation of the workload distribution policy because this is less than the specified LSF workload distribution of 64 licenses.

ENABLE_INTERACTIVE**Syntax**

ENABLE_INTERACTIVE=Y | N

Description

Globally enables one share of the licenses for interactive tasks.

Tip: Setting **ENABLE_INTERACTIVE** also allows **taskman** to detect and use batch resources.

Used for project mode only.

Default

Not defined. LSF License Scheduler allocates licenses equally to each cluster and does not distribute licenses for interactive tasks.

FAST_DISPATCH

Syntax

FAST_DISPATCH=Y | N

Description

Enables fast dispatch project mode for the license feature, which increases license utilization for project licenses.

Used for project mode only.

When enabled, LSF License Scheduler does not have to run **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** to verify that a license is free before each job dispatch. As soon as a job finishes, the cluster can reuse its licenses for another job of the same project, which keeps gaps between jobs small. However, because LSF License Scheduler does not run **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** to verify that the license is free, there is an increased chance of a license checkout failure for jobs if the license is already in use by a job in another project.

The fast dispatch project mode supports the following parameters in the Feature section:

- ALLOCATION
- DEMAND_LIMIT
- DISTRIBUTION
- GROUP_DISTRIBUTION
- LM_LICENSE_NAME
- LS_FEATURE_PERCENTAGE
- NAME
- NON_SHARED_DISTRIBUTION
- SERVICE_DOMAINS
- WORKLOAD_DISTRIBUTION

The fast dispatch project mode also supports the **MBD_HEARTBEAT_INTERVAL** parameter in the Parameters section.

Other parameters are not supported, including those that project mode supports, such as the following parameters:

- ACCINUSE_INCLUDES_OWNERSHIP
- DYNAMIC
- GROUP
- LOCAL_TO
- LS_ACTIVE_PERCENTAGE

Default

Not defined (N). LSF License Scheduler runs in project mode without fast dispatch.

HEARTBEAT_INTERVAL

Syntax

HEARTBEAT_INTERVAL=seconds

Description

The time interval between bld heartbeats indicating the bld is still running.

Default

60 seconds

HEARTBEAT_TIMEOUT

Syntax

HEARTBEAT_TIMEOUT=seconds

Description

The time a slave **bld** waits to hear from the master **bld** before assuming it has died.

Default

120 seconds

HIST_HOURS

Syntax

HIST_HOURS=hours

Description

Determines the rate of decay the accumulated use value used in fairshare and preemption decisions. When HIST_HOURS=0, accumulated use is not decayed.

Accumulated use is displayed by the **blstat** command under the heading ACUM_USE.

Used for project mode only.

Default

5 hours. Accumulated use decays to 1/10 of the original value over 5 hours.

HOSTS

Syntax

HOSTS=*host_name.domain_name* ...

Description

Defines LSF License Scheduler hosts, including candidate hosts.

Specify a fully qualified host name such as hostX.mycompany.com. You can omit the domain name if all your LSF License Scheduler clients run in the same DNS domain.

Used for both project mode and cluster mode.

INUSE_FROM_RUSAGE**Syntax**

INUSE_FROM_RUSAGE=Y | N

Description

When not defined or set to N, the **INUSE** value uses rusage from **bsub** job submissions merged with license checkout data reported by **blcollect** (as reported by **blstat**).

When INUSE_FROM_RUSAGE=Y, the **INUSE** value uses the rusage from **bsub** job submissions instead of waiting for the **blcollect** update. This can result in faster reallocation of tokens when using dynamic allocation (when **ALLOC_BUFFER** is set).

Used for cluster mode only.

Default

N

LIB_CONNTIMEOUT**Syntax**

LIB_CONNTIMEOUT=*seconds*

Description

Specifies a timeout value in seconds for communication between License Scheduler and LSF APIs. LIB_CONNTIMEOUT=0 indicates no timeout.

Used for both project mode and cluster mode.

Default

5 seconds

LIB_RECVMTIMEOUT**Syntax**

LIB_RECVMTIMEOUT=*seconds*

Description

Specifies a timeout value in seconds for communication between License Scheduler and LSF. LIB_RECVMTIMEOUT=0 indicates no timeout.

Used for both project mode and cluster mode.

Default

5 seconds

LM_REMOVE_INTERVAL**Syntax**

LM_REMOVE_INTERVAL=*seconds*

Description

Specifies the minimum time a job must have a license checked out before **lmremove** (for FlexNet) or **rlmremove** (for Reprise License Manager) can remove the license (using preemption). **lmremove** or **rlmremove** causes the license manager daemon and vendor daemons to close the TCP connection with the application.

LSF License Scheduler only considers preempting a job after this interval has elapsed.

When using **lmremove** or **rlmremove** as part of the preemption action (**LM_REMOVE_SUSP_JOBS**), define **LM_REMOVE_INTERVAL=0** to ensure that LSF License Scheduler can preempt a job immediately after checkout. After suspending the job, LSF License Scheduler then uses **lmremove** or **rlmremove** to release licenses from the job.

Used for both project mode and cluster mode.

Default

180 seconds

LM_REMOVE_SUSP_JOBS**Syntax**

LM_REMOVE_SUSP_JOBS=seconds

Description

Enables LSF License Scheduler to use **lmremove** (for FlexNet) or **rlmremove** (for Reprise License Manager) to remove license features from each recently-suspended job. After enabling this parameter, the preemption action is to suspend the job's processes and use **lmremove** or **rlmremove** to remove licenses from the application.

LSF License Scheduler continues to try removing the license feature for the specified number of seconds after the job is first suspended. When setting this parameter for an application, specify a value greater than the period following a license checkout that **lmremove** or **rlmremove** will fail for the application. This ensures that when a job suspends, its licenses are released. This period depends on the application.

When using **lmremove** or **rlmremove** as part of the preemption action, define **LM_REMOVE_INTERVAL=0** to ensure that LSF License Scheduler can preempt a job immediately after checkout. After suspending the job, LSF License Scheduler then uses **lmremove** or **rlmremove** to release licenses from the job.

This parameter applies to all features in fast dispatch project mode.

Used for fast dispatch project mode only.

Default

Undefined. The default preemption action is to send a TSTP signal to the job.

LM_REMOVE_SUSP_JOBS_INTERVAL**Syntax**

LM_REMOVE_SUSP_JOBS_INTERVAL=seconds

Description

Specifies the minimum length of time between subsequent child processes that LSF License Scheduler forks to run **lmremove** (for FlexNet) or **rlmremove** (for Reprise License Manager) every time it receives an update from a license collector daemon (**blcollect**).

Use this parameter when using **lmremove** or **rlmremove** as part of the preemption action (**LM_REMOVE_SUSP_JOBS**).

Used for fast dispatch project mode only.

Default

0. Uses the value of **LM_STAT_INTERVAL** instead.

LM_RESERVATION

Syntax

LM_RESERVATION=Y | N

Description

Enables LSF License Scheduler to support the FlexNet Manager license reservation keyword (**RESERVE**).

When LM_RESERVATION=Y is defined, LSF License Scheduler treats the **RESERVE** value in the FlexNet Manager license option file as **OTHERS** tokens instead of **FREE** tokens. The **RESERVE** value is now included in the **OTHERS** value in the **blstat** command output and is no longer included in the **FREE** value.

This parameter is ignored if it is defined in a time based configuration, or if the **WORKLOAD_DISTRIBUTION** parameter is defined in at least one feature.

Note: The license tokens that are reserved with FlexNet Manager must be used outside of the LSF License Scheduler cluster.

Default

N. The **RESERVE** value does not count as a used token.

LM_STAT_INTERVAL

Syntax

LM_STAT_INTERVAL=*seconds*

Description

Defines a time interval between calls that LSF License Scheduler makes to collect license usage information from the license manager.

Default

60 seconds

LM_STAT_TIMEOUT

Syntax

LM_STAT_TIMEOUT=*seconds*

Description

Sets the timeout value passed to the **lmutil lmstat**, **lmstat**, **rlmutil rlmstat**, or **rlmstat** command. The Parameters section setting is overwritten by the ServiceDomain setting, which is overwritten by the command line setting (**blcollect -t timeout**).

Used for both project mode and cluster mode.

Default

180 seconds

LM_TYPE

Syntax

LM_TYPE=FLEXLM | RLM

Description

Defines the license manager system that is used by the license servers. This determines how LSF License Scheduler communicates with the license servers.

Define LM_TYPE=FLEXLM if the license servers are using FlexNet Manager as the license manager system.

Define LM_TYPE=RLM if the license servers are using Reprise License Manager as the license manager system.

Default

FLEXLM

LMREMOVE_SUSP_JOBS (Obsolete)

Syntax

LMREMOVE_SUSP_JOBS=*seconds*

Description

Use **LM_REMOVE_SUSP_JOBS** instead. This parameter is only maintained for backwards compatibility.

LMREMOVE_SUSP_JOBS_INTERVAL (Obsolete)

Syntax

LMREMOVE_SUSP_JOBS_INTERVAL=*seconds*

Description

Replace **LMREMOVE_SUSP_JOBS_INTERVAL** with **LM_REMOVE_SUSP_JOBS_INTERVAL** instead. **LMREMOVE_SUSP_JOBS_INTERVAL** is only maintained for backwards compatibility.

LMSTAT_PATH

Syntax

LMSTAT_PATH=*path*

Description

Defines the full path to the location of the FlexNet command **lmutil** (or **lmstat**).

Used for project mode, fast dispatch project mode, and cluster mode.

LOG_EVENT

Syntax

LOG_EVENT=Y | N

Description

Enables logging of License Scheduler events in the `bld.stream` file.

Default

Not defined. Information is not logged.

LOG_INTERVAL**Syntax**

`LOG_INTERVAL=seconds`

Description

The interval between token allocation data logs in the data directory

Default

60 seconds

LS_DEBUG_BLC**Syntax**

`LS_DEBUG_BLC=log_class`

Description

Sets the debugging log class for the LSF License Scheduler **blcollect** daemon.

Used for both project mode and cluster mode.

Specifies the log class filtering to be applied to **blcollect**. Only messages belonging to the specified log class are recorded.

LS_DEBUG_BLC sets the log class and is used in combination with **LS_LOG_MASK**, which sets the log level. For example:

```
LS_LOG_MASK=LOG_DEBUG LS_DEBUG_BLC="LC_TRACE"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LS_DEBUG_BLC="LC_TRACE"
```

You need to restart the **blcollect** daemons after setting **LS_DEBUG_BLC** for your changes to take effect.

Valid values

Valid log classes are:

- LC_AUTH and LC2_AUTH: Log authentication messages
- LC_COMM and LC2_COMM: Log communication messages
- LC_FLEX - Log everything related to FLEX_STAT or FLEX_EXEC Flexera APIs
- LC_PERFM and LC2_PERFM: Log performance messages
- LC_PREEMPT - Log license preemption policy messages
- LC_RESREQ and LC2_RESREQ: Log resource requirement messages
- LC_SYS and LC2_SYS: Log system call messages
- LC_TRACE and LC2_TRACE: Log significant program walk steps

- LC_XDR and LC2_XDR: Log everything transferred by XDR

Default

Not defined.

LS_DEBUG_BLD

Syntax

```
LS_DEBUG_BLD=log_class
```

Description

Sets the debugging log class for the LSF License Scheduler bld daemon.

Used for both project mode and cluster mode.

Specifies the log class filtering to be applied to **bld**. Messages belonging to the specified log class are recorded. Not all debug message are controlled by log class.

LS_DEBUG_BLD sets the log class and is used in combination with MASK, which sets the log level. For example:

```
LS_LOG_MASK=LOG_DEBUG LS_DEBUG_BLD="LC_TRACE"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LS_DEBUG_BLD="LC_TRACE"
```

You need to restart the **bld** daemon after setting **LS_DEBUG_BLD** for your changes to take effect.

If you use the command **bladmin blddebug** to temporarily change this parameter without changing `lsf.licensescheduler`, you do not need to restart the daemons.

Valid values

Valid log classes are:

- LC_AUTH and LC2_AUTH: Log authentication messages
- LC_COMM and LC2_COMM: Log communication messages
- LC_FLEX - Log everything related to FLEX_STAT or FLEX_EXEC Flexera APIs
- LC_MEMORY - Log memory use messages
- LC_PREEMPT - Log license preemption policy messages
- LC_RESREQ and LC2_RESREQ: Log resource requirement messages
- LC_TRACE and LC2_TRACE: Log significant program walk steps
- LC_XDR and LC2_XDR: Log everything transferred by XDR

Valid values

Valid log classes are the same as for **LS_DEBUG_CMD**.

Default

Not defined.

LS_ENABLE_MAX_PREEMPT

Syntax

```
LS_ENABLE_MAX_PREEMPT=Y | N
```

Description

Enables maximum preemption time checking for LSF and **taskman** jobs.

When **LS_ENABLE_MAX_PREEMPT** is disabled, preemption times for **taskman** jobs are not checked regardless of the value of parameters **LS_MAX_TASKMAN_PREEMPT** in `lsf.licensescheduler` and **MAX_JOB_PREEMPT** in `lsb.queues`, `lsb.applications`, or `lsb.params`.

Used for project mode only.

Default

N

LS_LOG_MASK**Syntax**

`LS_LOG_MASK=message_log_level`

Description

Specifies the logging level of error messages for LSF License Scheduler daemons. If **LS_LOG_MASK** is not defined in `lsf.licensescheduler`, the value of **LSF_LOG_MASK** in `lsf.conf` is used. If neither **LS_LOG_MASK** nor **LSF_LOG_MASK** is defined, the default is **LOG_WARNING**.

Used for both project mode and cluster mode.

For example:

```
LS_LOG_MASK=LOG_DEBUG
```

The log levels in order from highest to lowest are:

- LOG_ERR
- LOG_WARNING
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

The most important LSF License Scheduler log messages are at the LOG_WARNING level. Messages at the LOG_DEBUG level are only useful for debugging.

Although message log level implements similar functionality to UNIX `syslog`, there is no dependency on UNIX `syslog`. It works even if messages are being logged to files instead of `syslog`.

LSF License Scheduler logs error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by **LS_LOG_MASK** determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used for basic debugging. The level LOG_DEBUG3 records all debugging messages, and can cause log files to grow very large; it is not often used. Most debugging is done at the level LOG_DEBUG2.

Default

LOG_WARNING

LS_MAX_STREAM_FILE_NUMBER

Syntax

LS_MAX_STREAM_FILE_NUMBER=*integer*

Description

Sets the number of saved `bld.stream.time_stamp` log files. When `LS_MAX_STREAM_FILE_NUMBER=2`, for example, the two most recent files are kept along with the current `bld.stream` file.

Used for both project mode and cluster mode.

Default

0 (old `bld.stream` file is not saved)

LS_MAX_STREAM_SIZE

Syntax

LS_MAX_STREAM_SIZE=*integer*

Description

Defines the maximum size of the `bld.stream` file in MB. Once this size is reached an **EVENT_END_OF_STREAM** is logged, a new `bld.stream` file is created, and the old `bld.stream` file is renamed `bld.stream.time_stamp`.

Used for both project mode and cluster mode.

Default

1024

LS_MAX_TASKMAN_PREEMPT

Syntax

LS_MAX_TASKMAN_PREEMPT=*integer*

Description

Defines the maximum number of times **taskman** jobs can be preempted.

Maximum preemption time checking for all jobs is enabled by **LS_ENABLE_MAX_PREEMPT**.

Used for project mode only.

Default

unlimited

LS_MAX_TASKMAN_SESSIONS

Syntax

LS_MAX_TASKMAN_SESSIONS=*integer*

Description

Defines the maximum number of **taskman** jobs that run simultaneously. This prevents system-wide performance issues that occur if there are a large number of **taskman** jobs running in License Scheduler.

The number of taskman sessions must be a positive integer.

The actual maximum number of taskman jobs is affected by the operating system file descriptor limit. Make sure the operating system file descriptor limit and the maximum concurrent connections are large enough to support all **taskman** tasks, License Scheduler (**bl***) commands, and connections between LSF License Scheduler and LSF.

Used for both project mode and cluster mode.

LS_PREEMPT_PEER**Syntax**

LS_PREEMPT_PEER=Y | N

Description

Enables bottom-up license token preemption in hierarchical project group configuration. LSF License Scheduler attempts to preempt tokens from the closest projects in the hierarchy first. This balances token ownership from the bottom up.

Used for project mode only.

Default

Not defined. Token preemption in hierarchical project groups is top down.

LS_STREAM_FILE**Syntax**

LS_STREAM_FILE=*path*

Used for both project mode and cluster mode.

Description

Defines the full path and filename of the **bld** event log file, `bld.stream` by default.

Default

LSF_TOP/work/db/bld.stream

MBD_HEARTBEAT_INTERVAL**Syntax**

MBD_HEARTBEAT_INTERVAL=*seconds*

Description

Sets the length of time the cluster license allocation remains unchanged after a cluster has disconnected from **bld**. After **MBD_HEARTBEAT_INTERVAL** has passed, the allocation is set to zero and licenses are redistributed to other clusters.

Used for cluster mode and fast dispatch project mode only.

Default

900 seconds

MBD_REFRESH_INTERVAL

Syntax

MBD_REFRESH_INTERVAL=*seconds*

Description

This parameter allows the administrator to independently control the minimum interval between load updates from **bld** and from LIM. **MBD_REFRESH_INTERVAL** controls the frequency of scheduling interactive (taskman) jobs and is read by the **mbatchd** daemon on startup. When **MBD_REFRESH_INTERVAL** is set or changed, you must restart the **bld** and **mbatchd** daemons in each cluster.

Used for both project mode and cluster mode.

Default

15 seconds

MERGE_BY_SERVICE_DOMAIN

Syntax

MERGE_BY_SERVICE_DOMAIN=Y | N

Description

If enabled, correlates job license checkout with the **lmutil lmstat, lmstat, rlmutil rlmstat, or rlmstat** output across all service domains first before reserving licenses.

In project mode (but not fast dispatch project mode), this parameter supports the case where the checkout license number for the application is less than or equal to the job rusage. If the checked out licenses are greater than the job rusage, the **ENABLE_DYNAMIC_RUSAGE** parameter is still required.

Default

N (Does not correlate job license checkout with the **lmutil, lmstat, rlmutil, or rlmstat** output across all service domains before reserving licenses)

PEAK_INUSE_PERIOD

Syntax

PEAK_INUSE_PERIOD=*seconds*

Description

Defines the interval over which a peak **INUSE** value is determined for dynamic license allocation in cluster mode for all license features over all service domains.

When defining the interval for LSF Advanced Edition submission clusters, the interval is determined for the entire LSF Advanced Edition mega-cluster (the submission cluster and its execution clusters).

Used for cluster mode only.

When defined in both the Parameters section and the Feature section, the Feature section definition is used for that license feature.

Default

300 seconds

PORT**Syntax**

PORT=*integer*

Description

Defines the TCP listening port used by LSF License Scheduler hosts, including candidate License Scheduler hosts. Specify any non-privileged port number.

Used for both project mode and cluster mode.

PREEMPT_ACTION**Syntax**

PREEMPT_ACTION=*action*

Description

Specifies the action used for taskman job preemption.

By default, if **PREEMPT_ACTION** is not configured, bld sends a TSTP signal to preempt taskman jobs.

You can specify a script using this parameter. For example, PREEMPT_ACTION = /home/user1/preempt.s runs **preempt.s** when preempting a taskman job.

Used for project mode only.

Default

Not defined. A TSTP signal is used to preempt taskman jobs.

PROJECT_GROUP_PATH**Syntax**

PROJECT_GROUP_PATH=Y | N

Description

Enables hierarchical project group paths for fast dispatch project mode, which enables the following:

- Features can use hierarchical project groups with project and project group names that are not unique, as long as the projects or project groups do not have the same parent. That is, you can define projects and project groups in more than one hierarchical project group.
- When specifying -Lp *license_project*, you can use paths to describe the project hierarchy without specifying the root group.

For example, if you have `root` as your root group, which has a child project group named `groupA` with a project named `proj1`, you can use `-Lp /groupA/proj1` to specify this project.

- Hierarchical project groups have a default project named `others` with a default share value of 0. Any projects that do not match the defined projects in a project group are assigned into the `others` project.

If there is already a project named `others`, the preexisting `others` project specification overrides the default project.

If **LSF_LIC_SCHED_STRICT_PROJECT_NAME** (in `lsf.conf`) and **PROJECT_GROUP_PATH** are both defined, **PROJECT_GROUP_PATH** takes precedence and overrides the **LSF_LIC_SCHED_STRICT_PROJECT_NAME** behavior for fast dispatch project mode.

Note: To use **PROJECT_GROUP_PATH**, you need LSF, Version 9.1.1, or later.

Used for fast dispatch project mode only.

Default

Not defined (N).

REMOTE_LMSTAT_PROTOCOL**Syntax**

```
REMOTE_LMSTAT_PROTOCOL=ssh [ssh_command_options] | rsh [rsh_command_options] | lsrun  
[lsrun_command_options]
```

Description

Specifies the method that LSF License Scheduler uses to connect to the remote agent host if there are remote license servers that need a remote agent host to collect license information.

If there are remote license servers that need a remote agent host to collect license information, LSF License Scheduler uses the specified command (and optional command options) to connect to the agent host. LSF License Scheduler automatically appends the name of the remote agent host to the command, so there is no need to specify the host with the command.

Note: LSF License Scheduler does not validate the specified command, so you must ensure that you correctly specify the command. The **blcollect** log file notes that the command failed, but not any details on the connection error. To determine specific connection errors, manually specify the command to connect to the remote server before specifying it in **REMOTE_LMSTAT_PROTOCOL**.

If using **lsrun** as the connection method, the remote agent host must be a server host in the LSF cluster and RES must be started on this host. If using **ssh** or **rsh** as the connection method, the agent host does not have to be a server host in the LSF cluster.

REMOTE_LMSTAT_PROTOCOL works with **REMOTE_LMSTAT_SERVERS**, which defines the remote license servers and remote agent hosts. If you do not define **REMOTE_LMSTAT_SERVERS**, **REMOTE_LMSTAT_PROTOCOL** is not used.

Used for both project mode and cluster mode.

Default

ssh

RLMSTAT_PATH**Syntax**

```
RLMSTAT_PATH=path
```

Description

Defines the full path to the location of the Reprise License Manager commands.

Used for both project mode and cluster mode.

Default

If not defined, this is set to **LMSTAT_PATH**.

STANDBY_CONNTIMEOUT**Syntax**

STANDBY_CONNTIMEOUT=*seconds*

Description

Sets the connection timeout the standby bld waits when trying to contact each host before assuming the host is unavailable.

Used for both project mode and cluster mode.

Default

5 seconds

Clusters section**Description**

Required. Lists the clusters that can use License Scheduler.

When configuring clusters for a WAN, the Clusters section of the master cluster must define its slave clusters.

The Clusters section is the same for both project mode and cluster mode.

Clusters section structure

The Clusters section begins and ends with the lines `Begin Clusters` and `End Clusters`. The second line is the column heading, `CLUSTERS`. Subsequent lines list participating clusters, one name per line:

```
Begin Clusters
CLUSTERS
cluster1
cluster2
End Clusters
```

CLUSTERS

Defines the name of each participating LSF cluster. Specify using one name per line.

ServiceDomain section**Description**

Required. Defines License Scheduler service domains as groups of physical license server hosts that serve a specific network.

The ServiceDomain section is the same for both project mode and cluster mode.

ServiceDomain section structure

Define a section for each License Scheduler service domain.

This example shows the structure of the section:

```
Begin ServiceDomain
NAME=DesignCenterB
LIC_SERVERS=((1888@hostD)(1888@hostE))
LIC_COLLECTOR=CenterB
End ServiceDomain
```

Parameters

- LIC_SERVERS
- LIC_COLLECTOR
- LM_STAT_INTERVAL
- LM_STAT_TIMEOUT
- LM_TYPE
- NAME
- REMOTE_LMSTAT_SERVERS

LIC_SERVERS**Syntax**

When using FlexNet as the license manager (LM_TYPE=FLEXLM): LIC_SERVERS=([(host_name | port_number@host_name | (port_number@host_name port_number@host_name port_number@host_name))] ...)

When using Reprise License Manager as the license manager (LM_TYPE=RLM): LIC_SERVERS=([(port_number@host_name | (port_number@host_name port_number@host_name port_number@host_name))] ...)

Description

Defines the license server hosts that make up the LSF License Scheduler service domain. Specify one or more license server hosts, and for each license server host, specify the number of the port that the license manager uses, then the at symbol (@), then the name of the host. Put one set of parentheses around the list, and one more set of parentheses around each host, unless you have redundant servers (three hosts sharing one license file). If you have redundant servers, the parentheses enclose all three hosts.

If LSF License Scheduler is using FlexNet as the license manager (that is, LM_TYPE=FLEXLM), and FlexNet uses the default port on a host, you can specify the host name without the port number.

If LSF License Scheduler is using Reprise License Manager as the license manager (that is, LM_TYPE=RLM), you must specify a port number for every license server host.

Used for both project mode and cluster mode.

Examples

- One FlexNet license server host:

```
LIC_SERVERS=((1700@hostA))
```

- Multiple license server hosts with unique license.dat files:

```
LIC_SERVERS=((1700@hostA) (1700@hostB) (1700@hostC))
```

- Redundant license server hosts sharing the same license.dat file:

```
LIC_SERVERS=((1700@hostD 1700@hostE 1700@hostF))
```

LIC_COLLECTOR**Syntax**

LIC_COLLECTOR=*license_collector_name*

Description

Defines a name for the license collector daemon (**blcollect**) to use in each service domain. **blcollect** collects license usage information from FlexNet and passes it to the License Scheduler daemon (**bld**). It improves performance by allowing you to distribute license information queries on multiple hosts.

You can only specify one collector per service domain, but you can specify one collector to serve multiple service domains. Each time you run **blcollect**, you must specify the name of the collector for the service domain. You can use any name you want.

Used for both project mode and cluster mode.

Default

Undefined. The License Scheduler daemon uses one license collector daemon for the entire cluster.

LM_STAT_INTERVAL**Syntax**

LM_STAT_INTERVAL=*seconds*

Description

Defines a time interval between calls that License Scheduler makes to collect license usage information from the license manager.

The value specified for a service domain overrides the global value defined in the Parameters section. Each service domain definition can specify a different value for this parameter.

Used for both project mode and cluster mode.

Default

License Scheduler applies the global value defined in the Parameters section.

LM_STAT_TIMEOUT**Syntax**

LM_STAT_TIMEOUT=*seconds*

Description

Sets the timeout value passed to the **lmutil lmstat**, **lmstat**, **rlmutil rlmstat**, or **rlmstat** command. The Parameters section setting is overwritten by the ServiceDomain setting, which is overwritten by the command line setting (**blcollect -t timeout**).

When using Reprise License Manager as the license manager (LM_TYPE=RLM), this parameter is ignored.

Used for both project mode and cluster mode.

Default

180 seconds

LM_TYPE**Syntax**

LM_TYPE=FLEXLM | RLM

Description

Defines the license manager system that is used by the license servers. This determines how LSF License Scheduler communicates with the license servers that are defined by the **LIC_SERVERS** parameter.

Define LM_TYPE=FLEXLM if the license servers are using FlexNet Manager as the license manager system.

Define LM_TYPE=RLM if the license servers are using Reprise License Manager as the license manager system. When LM_TYPE=RLM is defined, **LIC_SERVERS** must define *port_number@host_name* (that is, **LIC_SERVERS** must define a port number). Defining just the host name (or *@host_name*) without the port number is not allowed.

Default

FLEXLM

NAME

Defines the name of the service domain.

Used for both project mode and cluster mode.

REMOTE_LMSTAT_SERVERS**Syntax**

```
REMOTE_LMSTAT_SERVERS=host_name[(host_name ...)] [host_name[(host_name ...)] ...]
```

Description

Defines the remote license servers and, optionally, the remote agent hosts that serve these remote license servers.

A remote license server is a license server that does not run on the same domain as the license collector. A remote agent host serves remote license servers within the same domain, allowing the license collector to get license information on the remote license servers with a single remote connection.

Defining remote agent hosts are useful when there are both local and remote license servers because it is slower for the license collector to connect to multiple remote license servers to get license information than it is to connect to local license servers. The license collector connects to the remote agent host (using the command specified by the **REMOTE_LMSTAT_PROTOCOL** parameter) and calls **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** to collect license information from the license servers that the agent hosts serve. This allows the license collector to connect to one remote agent host to get license information from all the remote license servers on the same domain as the remote agent host. These license servers should be in the same subnet as the agent host to improve access.

Remote license servers must also be license servers defined in **LIC_SERVERS**. Any remote license servers defined in **REMOTE_LMSTAT_SERVERS** that are not also defined in **LIC_SERVERS** are ignored. Remote agent hosts that serve other license servers do not need to be defined in **LIC_SERVERS**. Remote agent hosts that are not defined in **LIC_SERVERS** function only as remote agents and not as license servers.

If you specify a remote agent host without additional servers (that is, the remote agent host does not serve any license servers), the remote agent host is considered to be a remote license server with itself as the remote agent host. That is, the license collector connects to the remote agent host and only gets license information on the remote agent host. Because these hosts are remote license servers, these remote agent hosts must also be defined as license servers in **LIC_SERVERS**, or they will be ignored.

Used for both project mode and cluster mode.

Examples

- One local license server (hostA) and one remote license server (hostB):

```
LIC_SERVERS=((1700@hostA) (1700@hostB))
REMOTE_LMSTAT_SERVERS=hostB
```

- The license collector runs **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** directly on hostA to get license information on hostA.
 - Because hostB is defined without additional license servers, hostB is a remote agent host that only serves itself. The license collector connects to hostB (using the command specified by the **REMOTE_LMSTAT_PROTOCOL** parameter) and runs **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** to get license information on 1700@hostB.
- One local license server (hostA), one remote agent host (hostB) that serves one remote license server (hostC), and one remote agent host (hostD) that serves two remote license servers (hostE and hostF):

```
LIC_SERVERS=((1700@hostA) (1700@hostB) (1700@hostC) (1700@hostD) (1700@hostE) (1700@hostF))
REMOTE_LMSTAT_SERVERS=hostB(hostC) hostD(hostE hostF)
```

- The license collector runs **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** directly to get license information from 1700@hostA, 1700@hostB, and 1700@hostD.
 - The license collector connects to hostB (using the command specified by the **REMOTE_LMSTAT_PROTOCOL** parameter) and runs **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** to get license information on 1700@hostC.
hostB and hostC should be in the same subnet to improve access.
 - The license collector connects to hostD (using the command specified by the **REMOTE_LMSTAT_PROTOCOL** parameter) and runs **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** to get license information on 1700@hostE and 1700@hostF.
hostD, hostE, and hostF should be in the same subnet to improve access.
- One local license server (hostA), one remote license server (hostB), and one remote agent host (hostC) that serves two remote license servers (hostD and hostE):

```
LIC_SERVERS=((1700@hostA) (1700@hostB) (1700@hostC) (1700@hostD) (1700@hostE))
REMOTE_LMSTAT_SERVERS=hostB hostC(hostD hostE)
```

- The license collector runs **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** directly to get license information on 1700@hostA and 1700@hostC.
- The license collector connects to hostB (using the command specified by the **REMOTE_LMSTAT_PROTOCOL** parameter) and runs **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** to get license information on 1700@hostB.
- The license collector connects to hostC (using the command specified by the **REMOTE_LMSTAT_PROTOCOL** parameter) and runs **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** to get license information on 1700@hostD and 1700@hostE.
hostC, hostD, and hostE should be in the same subnet to improve access.

Feature section

Description

Required. Defines license distribution policies.

Feature section structure

Define a section for each feature managed by License Scheduler. If there is a parameter with the same name in the Parameters section, setting these parameters in the Feature section overrides the global setting in the Parameters section.

```
Begin Feature
NAME=vcs
LM_LICENSE_NAME=vcs
...
Distribution policy
Parameters
...
End Feature
```

Parameters

- ACCINUSE_INCLUDES_OWNERSHIP
- ALLOC_BUFFER
- ALLOCATION
- CHECKOUT_FROM_FIRST_HOST_ONLY
- CLUSTER_DISTRIBUTION
- CLUSTER_MODE
- DEMAND_LIMIT
- DISABLE_PREEMPTION
- DISTRIBUTION
- DYNAMIC
- ENABLE_DYNAMIC_RUSAGE
- ENABLE_MINJOB_PREEMPTION
- FAST_DISPATCH
- FEATURE_DELTA
- FLEX_NAME
- GROUP
- GROUP_DISTRIBUTION
- INUSE_FROM_RUSAGE
- LM_LICENSE_NAME
- LM_REMOVE_INTERVAL
- LM_REMOVE_SUSP_JOBS
- LM_RESERVATION
- LMREMOVE_SUSP_JOBS
- LOCAL_TO
- LS_ACTIVE_PERCENTAGE
- LS_FEATURE_PERCENTAGE
- NAME
- NON_SHARED_DISTRIBUTION
- PEAK_INUSE_PERIOD
- PREEMPT_ORDER
- PREEMPT_RESERVE
- RETENTION_FACTOR

- SERVICE_DOMAINS
- WORKLOAD_DISTRIBUTION

ACCINUSE_INCLUDES_OWNERSHIP

Syntax

ACCINUSE_INCLUDES_OWNERSHIP=Y | N

Description

When not defined, accumulated use is incremented each scheduling cycle by (tokens in use) + (tokens reserved) if this exceeds the number of tokens owned.

When defined, accumulated use is incremented each scheduling cycle by (tokens in use) + (tokens reserved) regardless of the number of tokens owned.

This is useful for projects that have a very high ownership set when considered against the total number of tokens available for LSF workload. Projects can be starved for tokens when the ownership is set too high and this parameter is not set.

Accumulated use is displayed by the **blstat** command under the heading ACUM_USE.

Used for project mode only. Cluster mode and fast dispatch project mode do not track accumulated use.

Default

N, not enabled.

ALLOC_BUFFER

Syntax

ALLOC_BUFFER = *buffer* | *cluster_name buffer ... default buffer*

Description

Enables dynamic distribution of licenses across clusters in cluster mode.

Cluster names must be the names of clusters defined in the Clusters section of `lsf.licensescheduler`.

Used for cluster mode only.

ALLOC_BUFFER=*buffer* sets one buffer size for all clusters, while ALLOC_BUFFER=*cluster_name buffer ...* sets a different buffer size for each cluster.

The buffer size is used during dynamic redistribution of licenses. Increases in allocation are determined by the PEAK value, and increased by DEMAND for license tokens to a maximum increase of BUFFER, the buffer size configured by **ALLOC_BUFFER**. The license allocation can increase in steps as large as the buffer size, but no larger.

Allocation buffers help determine the maximum rate at which tokens can be transferred to a cluster as demand increases in the cluster. The maximum rate of transfer to a cluster is given by the allocation buffer divided by **MBD_REFRESH_INTERVAL**. Be careful not to set the allocation buffer too large so that licenses are not wasted because they are being allocated to a cluster that cannot use them.

Decreases in license allocation can be larger than the buffer size, but the allocation must remain at **PEAK + BUFFER** licenses. The license allocation includes up to the buffer size of extra licenses, in case demand increases.

Increasing the buffer size allows the license allocation to grow faster, but also increases the number of licenses that may go unused at any given time. The buffer value must be tuned for each license feature and cluster to balance these two objectives.

When defining the buffer size for LSF Advanced Edition submission clusters, the license allocation for the entire LSF Advanced Edition mega-cluster (the submission cluster and its execution clusters) can increase in steps as large as the buffer size, but no larger.

Detailed license distribution information is shown in the **blstat** output.

Use the keyword `default` to apply a buffer size to all remaining clusters. For example:

```
Begin Feature
NAME = f1
CLUSTER_DISTRIBUTION = WanServers(banff 1 berlin 1 boston 1)
ALLOC_BUFFER = banff 10 default 5
End Feature
```

In this example, dynamic distribution is enabled. The cluster `banff` has a buffer size of 10, and all remaining clusters have a buffer size of 5.

To allow a cluster to be able to use licenses only when another cluster does not need them, you can set the cluster distribution for the cluster to 0, and specify an allocation buffer for the number of tokens that the cluster can request.

For example:

```
Begin Feature
CLUSTER_DISTRIBUTION=Wan(CL1 0 CL2 1)
ALLOC_BUFFER=5
End Feature
```

When no jobs are running, the token allocation for `CL1` is 5. `CL1` can get more than 5 tokens if `CL2` does not require them.

Default

Not defined. Static distribution of licenses is used in cluster mode.

ALLOCATION

Syntax

`ALLOCATION=project_name (cluster_name [number_shares] ...) ...`

cluster_name

Specify LSF cluster names or interactive tasks that licenses are to be allocated to.

project_name

Specify a License Scheduler project (described in the Projects section or as default) that is allowed to use the licenses.

number_shares

Specify a positive integer representing the number of shares assigned to the cluster.

The number of shares assigned to a cluster is only meaningful when you compare it to the number assigned to other clusters. The total number of shares is the sum of the shares assigned to each cluster.

Description

Defines the allocation of license features across clusters and interactive tasks.

Used for project mode and fast dispatch project mode only.

`ALLOCATION` ignores the global setting of the `ENABLE_INTERACTIVE` parameter because `ALLOCATION` is configured for the license feature.

You can configure the allocation of license shares to:

- Change the share number between clusters for a feature

- Limit the scope of license usage and change the share number between LSF jobs and interactive tasks for a feature

When defining the allocation of license features for LSF Advanced Edition submission clusters, the allocation is for the entire LSF Advanced Edition mega-cluster (the submission cluster and its execution clusters).

Tip: To manage interactive tasks in License Scheduler projects, use the LSF Task Manager, `taskman`. The Task Manager utility is supported by License Scheduler.

Default

If `ENABLE_INTERACTIVE` is not set, each cluster receives equal share, and interactive tasks receive no shares.

Examples

Each example contains two clusters and 12 licenses of a specific feature.

Example 1

ALLOCATION is not configured. The `ENABLE_INTERACTIVE` parameter is not set.

```
Begin Parameters
...
ENABLE_INTERACTIVE=n
...
End Parameters
```

```
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1 (Lp1 1)
End Feature
```

Six licenses are allocated to each cluster. No licenses are allocated to interactive tasks.

Example 2

ALLOCATION is not configured. The `ENABLE_INTERACTIVE` parameter is set.

```
Begin Parameters
...
ENABLE_INTERACTIVE=y
...
End Parameters
```

```
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1 (Lp1 1)
End Feature
```

Four licenses are allocated to each cluster. Four licenses are allocated to interactive tasks.

Example 3

In the following example, the `ENABLE_INTERACTIVE` parameter does not affect the ALLOCATION configuration of the feature.

ALLOCATION is configured. The `ENABLE_INTERACTIVE` parameter is set.

```
Begin Parameters
...
ENABLE_INTERACTIVE=y
...
End Parameters
```

```

Begin Feature
NAME=ApplicationY
DISTRIBUTION=LicenseServer1 (Lp2 1)
ALLOCATION=Lp2(cluster1 1 cluster2 0 interactive 1)
End Feature

```

The ENABLE_INTERACTIVE setting is ignored. Licenses are shared equally between cluster1 and interactive tasks. Six licenses of ApplicationY are allocated to cluster1. Six licenses are allocated to interactive tasks.

Example 4

In the following example, the ENABLE_INTERACTIVE parameter does not affect the ALLOCATION configuration of the feature.

ALLOCATION is configured. The ENABLE_INTERACTIVE parameter is not set.

```

Begin Parameters
...
ENABLE_INTERACTIVE=n
...
End Parameters

```

```

Begin Feature
NAME=ApplicationZ
DISTRIBUTION=LicenseServer1 (Lp1 1)
ALLOCATION=Lp1(cluster1 0 cluster2 1 interactive 2)
End Feature

```

The ENABLE_INTERACTIVE setting is ignored. Four licenses of ApplicationZ are allocated to cluster2. Eight licenses are allocated to interactive tasks.

CHECKOUT_FROM_FIRST_HOST_ONLY

Syntax

CHECKOUT_FROM_FIRST_HOST_ONLY=Y | N

Description

If enabled, LSF License Scheduler only considers user@host information for the first execution host of a parallel job when merging the license usage data. Setting in individual Feature sections overrides the global setting in the Parameters section.

If a feature has multiple Feature sections (using **LOCAL_TO**), each section must have the same setting for **CHECKOUT_FROM_FIRST_HOST_ONLY**.

If disabled, LSF License Scheduler attempts to check out user@host keys in the parallel job constructed using the user name and all execution host names, and merges the corresponding checkout information on the service domain if found. If MERGE_BY_SERVICE_DOMAIN=Y is defined, LSF License Scheduler also merges multiple user@host data for parallel jobs across different service domains.

Default

Undefined (N). LSF License Scheduler attempts to check out user@host keys in the parallel job constructed using the user name and all execution host names, and merges the corresponding checkout information on the service domain if found.

CLUSTER_DISTRIBUTION

Syntax

CLUSTER_DISTRIBUTION=service_domain(cluster shares/min/max ...)...

service_domain

Specify a License Scheduler WAN service domain (described in the ServiceDomain section) that distributes licenses to multiple clusters, and the share for each cluster.

Specify a License Scheduler LAN service domain for a single cluster.

cluster

Specify each LSF cluster that accesses licenses from this service domain.

shares

For each cluster specified for a WAN service domain, specify a positive integer representing the number of shares assigned to the cluster. (Not required for a LAN service domain.)

The number of shares assigned to a cluster is only meaningful when you compare it to the number assigned to other clusters, or to the total number assigned by the service domain. The total number of shares is the sum of the shares assigned to each cluster.

min

The minimum allocation is allocated exclusively to the cluster, and is similar to the non-shared allocation in project mode.

Cluster shares take precedence over minimum allocations configured. If the minimum allocation exceeds the cluster's share of the total tokens, a cluster's allocation as given by **bold** may be less than the configured minimum allocation.

max

Optionally, specify a positive integer representing the maximum number of license tokens allocated to the cluster when dynamic allocation is enabled for a WAN service domain (when **ALLOC_BUFFER** is defined for the feature).

Description

CLUSTER_DISTRIBUTION must be defined when using cluster mode.

Defines the cross-cluster distribution policies for the license. The name of each service domain is followed by its distribution policy, in parentheses. The distribution policy determines how the licenses available in each service domain are distributed among the clients.

The distribution policy is a space-separated list with each cluster name followed by its share assignment. The share assignment determines what fraction of available licenses is assigned to each cluster, in the event of competition between clusters.

Examples

```
CLUSTER_DISTRIBUTION=wanserver(C11 1 C12 1 C13 1 C14 1)
```

```
CLUSTER_DISTRIBUTION = SD(C1 1 C2 1) SD1(C3 1 C4 1) SD2(C1 1) SD3(C2 1)
```

In these examples, wanserver, SD, and SD1 are WAN service domains, while SD2 and SD3 are LAN service domains serving a single cluster.

CLUSTER_MODE**Syntax**

```
CLUSTER_MODE=Y | N
```

Description

Enables cluster mode (instead of project mode) for the license feature.

Cluster mode emphasizes high utilization of license tokens above other considerations such as ownership. License ownership and sharing can still be configured, but within each cluster instead of across multiple clusters. Preemption of jobs (and licenses) also occurs within each cluster instead of across clusters.

Cluster mode was introduced in License Scheduler 8.0. Before cluster mode was introduced, project mode was the only choice available.

Default

Undefined (N). License Scheduler runs in project mode.

DEMAND_LIMIT

Syntax

DEMAND_LIMIT=*integer*

Description

Sets a limit to which LSF License Scheduler considers the demand by each project in each cluster when allocating licenses.

Used for fast dispatch project mode only.

When enabled, the demand limit helps prevent LSF License Scheduler from allocating more licenses to a project than can actually be used, which reduces license waste by limiting the demand that LSF License Scheduler considers. This is useful in cases when other resource limits are reached, LSF License Scheduler allocates more tokens than IBM Spectrum LSF can actually use because jobs are still pending due to lack of other resources.

When disabled (that is, DEMAND_LIMIT=0 is set), LSF License Scheduler takes into account all the demand reported by each cluster when scheduling.

DEMAND_LIMIT does not affect the DEMAND that **blstat** displays. Instead, **blstat** displays the entire demand sent for a project from all clusters. For example, one cluster reports a demand of 15 for a project. Another cluster reports a demand of 20 for the same project. When LSF License Scheduler allocates licenses, it takes into account a demand of five from each cluster for the project and the DEMAND that **blstat** displays is 35.

Periodically, each cluster sends a demand for each project. This is calculated in a cluster for a project by summing up the rusage of all jobs of the project pending due to lack of licenses. Whether to count a job's rusage in the demand depends on the job's pending reason. In general, the demand reported by a cluster only represents a potential demand from the project. It does not take into account other resources that are required to start a job. For example, a demand for 100 licenses is reported for a project. However, if LSF License Scheduler allocates 100 licenses to the project, the project does not necessarily use all 100 licenses due to slot availability, limits, or other scheduling constraints.

In project mode and fast dispatch project mode, **mbatchd** in each cluster sends a demand for licenses from each project. In project mode, LSF License Scheduler assumes that each project can actually use the demand that is sent to it. In fast dispatch project mode, **DEMAND_LIMIT** limits the amount of demand from each project in each cluster that is considered when scheduling.

Default

5

DISABLE_PREEMPTION

Syntax

DISABLE_PREEMPTION=Y | y | N | n

Description

When set Y or y, **DISABLE_PREEMPTION** prevents license preemption and resets the project ownership to 0. This ensures that license preemption never occurs regardless of whether projects are configured with ownership or NON_SHARED.

Used for project mode and fast dispatch mode only. Cluster mode does not support this parameter.

Default

N

DISTRIBUTION**Syntax**

DISTRIBUTION=[*service_domain_name*([*project_name number_shares*[/*number_licenses_owned*]] ... [default])] ...

service_domain_name

Specify a License Scheduler service domain (described in the ServiceDomain section) that distributes the licenses.

project_name

Specify a License Scheduler project (described in the Projects section) that is allowed to use the licenses.

number_shares

Specify a positive integer representing the number of shares assigned to the project.

The number of shares assigned to a project is only meaningful when you compare it to the number assigned to other projects, or to the total number assigned by the service domain. The total number of shares is the sum of the shares assigned to each project.

number_licenses_owned

Specify a slash (/) and a positive integer representing the number of licenses that the project owns. When configured, preemption is enabled and owned licenses are reclaimed using preemption when there is unmet demand.

default

A reserved keyword that represents the default project if the job submission does not specify a project (**bsub -Lp**), or the specified project is not configured in the Projects section of `lsf.licensescheduler`. Jobs that belong to projects do not get a share of the tokens when the project is not explicitly defined in DISTRIBUTION.

Description

Used for project mode and fast dispatch project mode only.

One of **DISTRIBUTION** or **GROUP_DISTRIBUTION** must be defined when using project mode. **GROUP_DISTRIBUTION** and **DISTRIBUTION** are mutually exclusive. If defined in the same feature, the License Scheduler daemon returns an error and ignores this feature.

Defines the distribution policies for the license. The name of each service domain is followed by its distribution policy, in parentheses. The distribution policy determines how the licenses available in each service domain are distributed among the clients.

When in fast dispatch project mode, you can only specify one service domain.

The distribution policy is a space-separated list with each project name followed by its share assignment. The share assignment determines what fraction of available licenses is assigned to each project, in the

event of competition between projects. Optionally, the share assignment is followed by a slash and the number of licenses owned by that project. License ownership enables a preemption policy. In the event of competition between projects, projects that own licenses preempt jobs. Licenses are returned to the owner immediately.

Examples

```
DISTRIBUTION=wanserver (Lp1 1 Lp2 1 Lp3 1 Lp4 1)
```

In this example, the service domain named `wanserver` shares licenses equally among four projects. If all projects are competing for a total of eight licenses, each project is entitled to two licenses at all times. If all projects are competing for only two licenses in total, each project is entitled to a license half the time.

```
DISTRIBUTION=lanserver1 (Lp1 1 Lp2 2/6)
```

In this example, the service domain named `lanserver1` allows `Lp1` to use one third of the available licenses and `Lp2` can use two thirds of the licenses. However, `Lp2` is always entitled to six licenses, and can preempt another project to get the licenses immediately if they are needed. If the projects are competing for a total of 12 licenses, `Lp2` is entitled to eight licenses (six on demand, and two more as soon as they are free). If the projects are competing for only six licenses in total, `Lp2` is entitled to all of them, and `Lp1` can only use licenses when `Lp2` does not need them.

DYNAMIC

Syntax

```
DYNAMIC=Y | N
```

Description

If you specify `DYNAMIC=Y`, you must specify a duration in an `rusage` resource requirement for the feature. This enables License Scheduler to treat the license as a dynamic resource and prevents License Scheduler from scheduling tokens for the feature when they are not available, or reserving license tokens when they should actually be free.

Used for project mode only. Cluster mode and fast dispatch project mode do not support `rusage` duration.

ENABLE_DYNAMIC_RUSAGE

Syntax

```
ENABLE_DYNAMIC_RUSAGE=Y | N
```

Description

When set, `ENABLE_DYNAMIC_RUSAGE` enables all license checkouts for features where the job checks out licenses in excess of `rusage` to be considered managed checkouts, instead of unmanaged (or OTHERS).

Used for project mode only. Cluster mode and fast dispatch project mode do not support this parameter.

ENABLE_MINJOB_PREEMPTION

Syntax

```
ENABLE_MINJOB_PREEMPTION=Y | N
```

Description

Minimizes the overall number of preempted jobs by enabling job list optimization. For example, for a job that requires 10 licenses, License Scheduler preempts one job that uses 10 or more licenses rather than 10 jobs that each use one license.

Used for project mode only

Default

Undefined: License Scheduler does not optimize the job list when selecting jobs to preempt.

FAST_DISPATCH**Syntax**

FAST_DISPATCH=Y | N

Description

Enables fast dispatch project mode for the license feature, which increases license utilization for project licenses.

Used for project mode only.

When enabled, LSF License Scheduler does not have to run **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** to verify that a license is free before each job dispatch. As soon as a job finishes, the cluster can reuse its licenses for another job of the same project, which keeps gaps between jobs small. However, because LSF License Scheduler does not run **lmutil**, **lmstat**, **rlmutil**, or **rlmstat** to verify that the license is free, there is an increased chance of a license checkout failure for jobs if the license is already in use by a job in another project.

The fast dispatch project mode supports the following parameters in the Feature section:

- ALLOCATION
- DEMAND_LIMIT
- DISTRIBUTION
- GROUP_DISTRIBUTION
- LM_LICENSE_NAME
- LS_FEATURE_PERCENTAGE
- NAME
- NON_SHARED_DISTRIBUTION
- SERVICE_DOMAINS
- WORKLOAD_DISTRIBUTION

The fast dispatch project mode also supports the MBD_HEARTBEAT_INTERVAL parameter in the Parameters section.

Other parameters are not supported, including those that project mode supports, such as the following parameters:

- ACCINUSE_INCLUDES_OWNERSHIP
- DYNAMIC
- GROUP
- LOCAL_TO
- LS_ACTIVE_PERCENTAGE

Default

Undefined (N). License Scheduler runs in project mode without fast dispatch.

FEATURE_DELTA**Syntax**

FEATURE_DELTA=*service_domain_name*(abs | mult *number*) ...

FEATURE_DELTA=*service_domain_name*(abs dyn ([init([integer])] [max([integer])] [step([integer])]
[target([number])] [interval([integer])])

Description

Allows an administrator to specify a number of dispatched jobs greater than actual tokens managed by the license server. Use this feature when utilization is low due to an external application's slow checkout of licenses. The result is higher average utilization at the cost of running some jobs less efficiently (those jobs that get dispatched but have to wait for the token).

Used for both project mode and cluster mode.

Each service domain can only be listed once in the **FEATURE_DELTA** parameter, otherwise all **FEATURE_DELTA** settings are ignored.

- **abs**: Specifies an absolute number of dispatched jobs greater than tokens managed by the license server. Specify a positive integer.
- **mult**: Specifies a multiplication factor of dispatched jobs greater than tokens managed by the license server. Specify a decimal number between 1 and 2. If you specify a number greater than 2, most of your jobs wait for a token and run less efficiently.
- **dyn**: Use with the **abs** or **mult** keywords (instead of a number) to enable LSF License Scheduler to dynamically adjust the number of dispatched jobs greater than tokens managed by the license server.

Enable dynamic adjustment for features that have a fluctuating license utilization and therefore require a more flexible value. The following optional parameters control this dynamic adjustment:

- **init**: Specifies the initial number of dispatched jobs greater than tokens managed by the license server when the **bls** daemon starts up or there are no jobs running in the servers that are specified in the **FEATURE_DELTA** parameter. Specify an integer for **abs** or a decimal number for **mult**. The default value is 0 for **abs** and 1.0 for **mult**.
- **max**: Specifies the adjustment number to calculate the maximum number of dispatched jobs greater than tokens managed by the license server. To obtain the adjusted maximum number, the value of **max** is added to the **init** value for **abs** and multiplied by the **init** value for **mult**. Specify an integer for **abs** or a decimal number for **mult**.

The default value for **abs** is the same as the **init** value, and the default value for **mult** is 2.0. This means that the default adjusted maximum number is double the **init** value for both **abs** and **mult**.
- **step**: Specifies the amount to adjust (if needed) in each interval. Specify an integer for **abs** or a decimal number for **mult**. The default value is either 1 or 5% of the **init** value, whichever is greater.
- **dynstep** (*hist_time factor*): Use instead of **step** to specify a dynamic amount to adjust in each interval, where *hist_time* is the time window for LSF License Scheduler to calculate the average token utilization, in minutes and *factor* is the factor to adjust the dynamic amount to be more or less aggressive towards the target value.
- **target**: Specifies the target license utilization rate. Specify a decimal number. The default value is 0.9.
- **interval**: Specifies the license usage checking interval per collector update. That is, 1 indicates that LSF License Scheduler checks and adjusts every collector update, and 2 indicates that LSF License Scheduler checks and adjusts every 2 collector updates. The default value is 3 (LSF License Scheduler checks and adjusts every 3 collector updates).

Example

In `lsf.licensescheduler`, define the following Service domains and Feature section:

```
Begin ServiceDomain
NAME = LanServer
LIC_SERVERS = ((38000@b110e-15) (38000@b110e-16))
End ServiceDomain
```

```
Begin ServiceDomain
NAME = LanServer1
LIC_SERVERS = ((38000@b110e-17))
End ServiceDomain
```

```
Begin Feature
NAME = f1
DISTRIBUTION = LanServer(p1 1 p2 1 p3 1) LanServer1(p1 1 p2 1)
FEATURE_DELTA = LanServer(mult 1.5) LanServer1(abs 4)
End Feature
```

Result

If `LanServer` had 6 licenses and `LanServer1` had 5 licenses, the number of tokens are now:

`LanServer`: $(6 * 1.5) = 9$

`LanServer1`: $(5 + 4) = 9$

The output of **blstat** displays counts the total number, as added or multiplied by the value specified in **FEATURE_DELTA**.

FLEX_NAME (Obsolete)

Syntax

`FLEX_NAME=feature_name1 [feature_name2 ...]`

Description

Replace **FLEX_NAME** with **LM_LICENSE_NAME**. **FLEX_NAME** is only maintained for backwards compatibility.

GROUP

Syntax

`GROUP=[group_name(project_name...)] ...`

group_name

Specify a name for a group of projects. This is different from a `ProjectGroup` section; groups of projects are not hierarchical.

project_name

Specify a License Scheduler project (described in the `Projects` section) that is allowed to use the licenses. The project must appear in the `DISTRIBUTION` and only belong to one group.

Description

Defines groups of projects and specifies the name of each group. The groups defined here are used for group preemption. The number of licenses owned by the group is the total number of licenses owned by member projects.

Used for project mode only. Cluster mode and fast dispatch project mode do not support this parameter.

This parameter is ignored if `GROUP_DISTRIBUTION` is also defined.

Example

For example, without the GROUP configuration shown, proj1 owns 4 license tokens and can reclaim them using preemption. After adding the GROUP configuration, proj1 and proj2 together own 8 license tokens. If proj2 is idle, proj1 is able to reclaim all 8 license tokens using preemption.

```
Begin Feature
NAME = AppY
DISTRIBUTION = LanServer1(proj1 1/4 proj2 1/4 proj3 2)
GROUP = GroupA(proj1 proj2)
End Feature
```

GROUP_DISTRIBUTION

Syntax

GROUP_DISTRIBUTION=*top_level_hierarchy_name*

top_level_hierarchy_name

Specify the name of the top level hierarchical group.

Description

Defines the name of the hierarchical group containing the distribution policy attached to this feature, where the hierarchical distribution policy is defined in a ProjectGroup section.

One of **DISTRIBUTION** or **GROUP_DISTRIBUTION** must be defined when using project mode.

GROUP_DISTRIBUTION and **DISTRIBUTION** are mutually exclusive. If defined in the same feature, the License Scheduler daemon returns an error and ignores this feature.

If **GROUP** is also defined, it is ignored in favor of **GROUP_DISTRIBUTION**.

Example

The following example shows the **GROUP_DISTRIBUTION** parameter hierarchical scheduling for the top-level hierarchical group named groups. The **SERVICE_DOMAINS** parameter defines a list of service domains that provide tokens for the group.

```
Begin Feature
NAME = myjob2
GROUP_DISTRIBUTION = groups
SERVICE_DOMAINS = LanServer wanServer
End Feature
```

INUSE_FROM_RUSAGE

Syntax

INUSE_FROM_RUSAGE=Y | N

Description

When not defined or set to N, the **INUSE** value uses rusage from **bsub** job submissions merged with license checkout data reported by **blcollect** (as reported by **blstat**).

When INUSE_FROM_RUSAGE=Y, the **INUSE** value uses the rusage from **bsub** job submissions instead of waiting for the blcollect update. This can result in faster reallocation of tokens when using dynamic allocation (when **ALLOC_BUFFER** is set).

When used for individual license features, the Feature section setting overrides the global Parameters section setting.

Used for cluster mode only.

Default

N

LM_LICENSE_NAME**Syntax**LM_LICENSE_NAME=*feature_name1* [*feature_name2* ...]**Description**

Defines the feature name—the name used by the license manager to identify the type of license. You only need to specify this parameter if the License Scheduler token name is not identical to the license manager feature name.

LM_LICENSE_NAME allows the **NAME** parameter to be an alias of the license manager feature name. For feature names that start with a number or contain a dash (-), you must set both **NAME** and **LM_LICENSE_NAME**, where **LM_LICENSE_NAME** is the actual license manager feature name, and **NAME** is an arbitrary license token name you choose.

Specify a space-delimited list of feature names in **LM_LICENSE_NAME** to combine multiple license manager features into one feature name specified under the **NAME** parameter. This allows you to use the same feature name for multiple license manager features (that are interchangeable for applications). LSF recognizes the alias of the combined feature (specified in **NAME**) as a feature name instead of the individual license manager feature names specified in **LM_LICENSE_NAME**. When submitting a job to LSF, users specify the combined feature name in the **bsub** rusage string, which allows the job to use any token from any of the features specified in **LM_LICENSE_NAME**.

Example

To specify AppZ201 as an alias for the license manager feature named 201-AppZ:

```
Begin Feature
LM_LICENSE_NAME=201-AppZ
NAME=AppZ201
DISTRIBUTION=LanServer1(Lp1 1 Lp2 1)
End Feature
```

To combine two license manager features (201-AppZ and 202-AppZ) into a feature named AppZ201:

```
Begin Feature
LM_LICENSE_NAME=201-AppZ 202-AppZ
NAME=AppZ201
DISTRIBUTION=LanServer1(Lp1 1 Lp2 1)
End Feature
```

AppZ201 is a combined feature that uses both 201-AppZ and 202-AppZ tokens. Submitting a job with AppZ201 in the rusage string (for example, `bsub -Lp Lp1 -R "rusage[AppZ201=2]" myjob`) means that the job checks out tokens for either 201-AppZ or 202-AppZ.

LM_REMOVE_INTERVAL**Syntax**LM_REMOVE_INTERVAL=*seconds***Description**

Specifies the minimum time a job must have a license checked out before **lmremove** or **rlmremove** can remove the license. **lmremove** or **rlmremove** causes the license manager daemon and vendor daemons to close the TCP connection with the application. They can then retry the license checkout.

When using **lmremove** or **rlmremove** as part of the preemption action (**LM_REMOVE_SUSP_JOBS**), define **LM_REMOVE_INTERVAL=0** to ensure that LSF License Scheduler can preempt a job immediately after checkout. After suspending the job, LSF License Scheduler then uses **lmremove** or **rlmremove** to release licenses from the job.

Used for both project mode and cluster mode.

The value specified for a feature overrides the global value defined in the Parameters section. Each feature definition can specify a different value for this parameter.

Default

Undefined: LSF License Scheduler applies the global value.

LM_REMOVE_SUSP_JOBS

Syntax

LM_REMOVE_SUSP_JOBS=seconds

Description

Enables LSF License Scheduler to use **lmremove** or **rlmremove** to remove license features from each recently-suspended job. After enabling this parameter, the preemption action is to suspend the job's processes and use **lmremove** or **rlmremove** to remove licenses from the application. **lmremove** or **rlmremove** causes the license manager daemon and vendor daemons to close the TCP connection with the application.

LSF License Scheduler continues to try removing the license feature for the specified number of seconds after the job is first suspended. When setting this parameter for an application, specify a value greater than the period following a license checkout that **lmremove** or **rlmremove** will fail for the application. This ensures that when a job suspends, its licenses are released. This period depends on the application.

When using **lmremove** or **rlmremove** as part of the preemption action, define **LM_REMOVE_INTERVAL=0** to ensure that LSF License Scheduler can preempt a job immediately after checkout. After suspending the job, LSF License Scheduler then uses **lmremove** or **rlmremove** to release licenses from the job.

Used for fast dispatch project mode only.

The value specified for a feature overrides the global value defined in the Parameters section. Each feature definition can specify a different value for this parameter.

Default

Undefined. The default preemption action is to send a TSTP signal to the job.

LM_RESERVATION

Syntax

LM_RESERVATION=Y | N

Description

Enables LSF License Scheduler to support the FlexNet Manager license reservation keyword (**RESERVE**).

When **LM_RESERVATION=Y** is defined, LSF License Scheduler treats the **RESERVE** value in the FlexNet Manager license option file as **OTHERS** tokens instead of **FREE** tokens. The **RESERVE** value is now included in the **OTHERS** value in the **blstat** command output and is no longer included in the **FREE** value.

This parameter is ignored if it is defined in a time based configuration, or if the **WORKLOAD_DISTRIBUTION** parameter is defined in this feature.

The value specified for a feature overrides the global value defined in the Parameters section. Each feature definition can specify a different value for this parameter.

Note: The license tokens that are reserved with FlexNet Manager must be used outside of LSF License Scheduler.

Default

Undefined: LSF License Scheduler applies the global value.

LMREMOVE_SUSP_JOBS (Obsolete)

Syntax

LMREMOVE_SUSP_JOBS=*seconds*

Description

Replace **LMREMOVE_SUSP_JOBS** with **LM_REMOVE_SUSP_JOBS**. **LMREMOVE_SUSP_JOBS** is only maintained for backwards compatibility.

LOCAL_TO

Syntax

LOCAL_TO=*cluster_name* | *location_name* (*cluster_name* [*cluster_name* ...])

Description

Used for project mode only. Cluster mode and fast dispatch project mode do not support this parameter.

Configures token locality for the license feature. You must configure different feature sections for same feature based on their locality. By default, if **LOCAL_TO** is not defined, the feature is available to all clients and is not restricted by geographical location. When **LOCAL_TO** is configured, for a feature, License Scheduler treats license features served to different locations as different token names, and distributes the tokens to projects according the distribution and allocation policies for the feature.

LOCAL_TO cannot be defined for LSF Advanced Edition submission clusters.

LOCAL_TO allows you to limit features from different service domains to specific clusters, so License Scheduler only grants tokens of a feature to jobs from clusters that are entitled to them.

For example, if your license servers restrict the serving of license tokens to specific geographical locations, use **LOCAL_TO** to specify the locality of a license token if any feature cannot be shared across all the locations. This avoids having to define different distribution and allocation policies for different service domains, and allows hierarchical group configurations.

License Scheduler manages features with different localities as different resources. Use **blinfo** and **blstat** to see the different resource information for the features depending on their cluster locality.

License features with different localities must be defined in different feature sections. The same Service Domain can appear only once in the configuration for a given license feature.

A configuration like LOCAL_TO=Site1(*clusterA clusterB*) configures the feature for more than one cluster when using project mode.

A configuration like LOCAL_TO=*clusterA* configures locality for only one cluster. This is the same as LOCAL_TO=*clusterA*(*clusterA*).

Cluster names must be the names of clusters defined in the Clusters section of `lsf.licensescheduler`.

Examples

```
Begin Feature
NAME = hspice
DISTRIBUTION = SD1 (Lp1 1 Lp2 1)
LOCAL_TO = siteUS(clusterA clusterB)
End Feature
```

```
Begin Feature
NAME = hspice
DISTRIBUTION = SD2 (Lp1 1 Lp2 1)
LOCAL_TO = clusterA
End Feature
```

```
Begin Feature
NAME = hspice
DISTRIBUTION = SD3 (Lp1 1 Lp2 1) SD4 (Lp1 1 Lp2 1)
End Feature
```

Or use the hierarchical group configuration (**GROUP_DISTRIBUTION**):

```
Begin Feature
NAME = hspice
GROUP_DISTRIBUTION = group1
SERVICE_DOMAINS = SD1
LOCAL_TO = clusterA
End Feature
```

```
Begin Feature
NAME = hspice
GROUP_DISTRIBUTION = group1
SERVICE_DOMAINS = SD2
LOCAL_TO = clusterB
End Feature
```

```
Begin Feature
NAME = hspice
GROUP_DISTRIBUTION = group1
SERVICE_DOMAINS = SD3 SD4
End Feature
```

Default

Not defined. The feature is available to all clusters and taskman jobs, and is not restricted by cluster.

LS_ACTIVE_PERCENTAGE**Syntax**

LS_ACTIVE_PERCENTAGE=Y | N

Description

Configures license ownership in percentages instead of absolute numbers and adjusts ownership for inactive projects. Sets LS_FEATURE_PERCENTAGE=Y automatically.

Setting LS_ACTIVE_PERCENTAGE=Y dynamically adjusts ownership based on project activity, setting ownership to zero for inactive projects and restoring the configured ownership setting when projects become active. If the total ownership for the license feature is greater than 100%, each ownership value is scaled appropriately for a total ownership of 100%.

Used for project mode only. Cluster mode and fast dispatch project mode do not support this parameter.

Default

N (Ownership values are not changed based on project activity.)

LS_FEATURE_PERCENTAGE

Syntax

LS_FEATURE_PERCENTAGE=Y | N

Description

Configures license ownership in percentages instead of absolute numbers. When not combined with hierarchical projects, affects the owned values in DISTRIBUTION and the NON_SHARED_DISTRIBUTION values only.

When using hierarchical projects, percentage is applied to OWNERSHIP, LIMITS, and NON_SHARED values.

Used for project mode and fast dispatch project mode only.

For example:

```

Begin Feature
LS_FEATURE_PERCENTAGE = Y
DISTRIBUTION = LanServer (p1 1 p2 1 p3 1/20)
...
End Feature

```

The service domain LanServer shares licenses equally among three License Scheduler projects. P3 is always entitled to 20% of the total licenses, and can preempt another project to get the licenses immediately if they are needed.

Example 1

```

Begin Feature
LS_FEATURE_PERCENTAGE = Y
DISTRIBUTION = LanServer (p1 1 p2 1 p3 1/20)
...
End Feature

```

The service domain LanServer shares licenses equally among three License Scheduler projects. P3 is always entitled to 20% of the total licenses, and can preempt another project to get the licenses immediately if they are needed.

Example 2

With LS_FEATURE_PERCENTAGE=Y in feature section and using hierarchical project groups:

```

Begin ProjectGroup
GROUP      SHARES  OWNERSHIP  LIMITS  NON_SHARED
(R (A p4)) (1 1)    ()        ()      ()
(A (B p3)) (1 1)    (- 10)   (- 20)  ()
(B (p1 p2)) (1 1)   (30 -)   ()      (- 5)
End ProjectGroup

```

Project p1 owns 30% of the total license and project p3 owns 10% of total licenses. P3's LIMITS is 20% of total licenses, and p2's NON_SHARED is 5%.

Default

N (Ownership is not configured with percentages, but with absolute numbers.)

NAME

Syntax

NAME=*feature_name*

Required. Defines the token name—the name used by License Scheduler and LSF to identify the license feature.

Normally, license token names should be the same as the FlexNet Licensing feature names, as they represent the same license. However, LSF does not support names that start with a number, or names containing a dash or hyphen character (-), which may be used in the FlexNet Licensing feature name.

NON_SHARED_DISTRIBUTION

Syntax

`NON_SHARED_DISTRIBUTION=service_domain_name ([project_name number_non_shared_licenses] ...) ...`

service_domain_name

Specify a License Scheduler service domain (described in the ServiceDomain section) that distributes the licenses.

project_name

Specify a License Scheduler project (described in the section) that is allowed to use the licenses.

number_non_shared_licenses

Specify a positive integer representing the number of non-shared licenses that the project owns.

Description

Defines non-shared licenses. Non-shared licenses are privately owned, and are not shared with other license projects. They are available only to one project.

Used for project mode and fast dispatch project mode only.

Use **blinfo -a** to display **NON_SHARED_DISTRIBUTION** information.

For projects defined with **NON_SHARED_DISTRIBUTION**, you must assign the project **OWNERSHIP** an equal or greater number of tokens than the number of non-shared licenses. If the number of owned licenses is less than the number of non-shared licenses, **OWNERSHIP** is set to the number of non-shared licenses.

Examples

- If the number of tokens normally given to a project (to satisfy the **DISTRIBUTION** share ratio) is larger than its **NON_SHARED_DISTRIBUTION** value, the **DISTRIBUTION** share ratio takes effect first.

```
Begin Feature
NAME=f1 # total 15 on LanServer
LM_LICENSE_NAME=VCS-RUNTIME
DISTRIBUTION=LanServer(Lp1 4/10 Lp2 1)
NON_SHARED_DISTRIBUTION=LanServer(Lp1 10)
End Feature
```

In this example, 10 non-shared licenses are defined for the Lp1 project on LanServer. The **DISTRIBUTION** share ratio for Lp1:Lp2 is 4:1. If there are 15 licenses, Lp1 will normally get 12 licenses, which is larger than its **NON_SHARED_DISTRIBUTION** value of 10. Therefore, the **DISTRIBUTION** share ratio takes effect, so Lp1 gets 12 licenses and Lp2 gets 3 licenses for the 4:1 share ratio.

- If the number of tokens normally given to a project (to satisfy the **DISTRIBUTION** share ratio) is smaller than its **NON_SHARED_DISTRIBUTION** value, the project will first get the number of tokens equal to **NON_SHARED_DISTRIBUTION**, then the **DISTRIBUTION** share ratio for the other projects takes effect for the remaining licenses.

- For one project with non-shared licenses and one project with no non-shared licenses: , the project with no non-shared licenses is given all the remaining licenses since it would normally be given more according to the **DISTRIBUTION** share ratio:

```
Begin Feature
NAME=f1 # total 15 on LanServer
LM_LICENSE_NAME=VCS-RUNTIME
DISTRIBUTION=LanServer(Lp1 1/10 Lp2 4)
NON_SHARED_DISTRIBUTION=LanServer(Lp1 10)
End Feature
```

In this example, 10 non-shared licenses are defined for the Lp1 project on LanServer. The **DISTRIBUTION** share ratio for Lp1:Lp2 is 1:4. If there are 15 licenses, Lp1 will normally get three licenses, which is smaller than its **NON_SHARED_DISTRIBUTION** value of 10. Therefore, Lp1 gets the first 10 licenses, and Lp2 gets the remaining five licenses (since it would normally get more according to the share ratio).

- For one project with non-shared licenses and two or more projects with no non-shared licenses, the two projects with no non-shared licenses are assigned the remaining licenses according to the **DISTRIBUTION** share ratio with each other, ignoring the share ratio for the project with non-shared licenses.

```
Begin Feature
NAME=f1 # total 15 on LanServer
LM_LICENSE_NAME=VCS-RUNTIME
DISTRIBUTION=LanServer(Lp1 1/10 Lp2 4 Lp3 2)
NON_SHARED_DISTRIBUTION=LanServer(Lp1 10)
End Feature
```

In this example, 10 non-shared licenses are defined for the Lp1 project on LanServer. The **DISTRIBUTION** share ratio for Lp1:Lp2:Lp3 is 1:4:2. If there are 15 licenses, Lp1 will normally get two licenses, which is smaller than its **NON_SHARED_DISTRIBUTION** value of 10. Therefore, Lp1 gets the first 10 licenses. The remaining licenses are given to Lp2 and Lp3 to a ratio of 4:2, so Lp2 gets three licenses and Lp3 gets two licenses.

- For two projects with non-shared licenses and one with no non-shared licenses, the one project with no non-shared licenses is given the remaining licenses after the two projects are given their non-shared licenses:

```
Begin Feature
NAME=f1 # total 15 on LanServer
LM_LICENSE_NAME=VCS-RUNTIME
DISTRIBUTION=LanServer(Lp1 1/10 Lp2 4 Lp3 2/5)
NON_SHARED_DISTRIBUTION=LanServer(Lp1 10 Lp3 5)
End Feature
```

In this example, 10 non-shared licenses are defined for the Lp1 project and five non-shared license are defined for the Lp3 project on LanServer. The **DISTRIBUTION** share ratio for Lp1:Lp2:Lp3 is 1:4:2. If there are 15 licenses, Lp1 will normally get two licenses and Lp3 will normally get four licenses, which are both smaller than their corresponding **NON_SHARED_DISTRIBUTION** values. Therefore, Lp1 gets 10 licenses and Lp3 gets five licenses. Lp2 gets no licenses even though it normally has the largest share because Lp1 and Lp3 have non-shared licenses.

PEAK_INUSE_PERIOD

Syntax

PEAK_INUSE_PERIOD=*seconds* | *cluster seconds* ...

Description

Defines the interval over which a peak **INUSE** value is determined for dynamic license allocation in cluster mode for this license features and service domain.

Use keyword `default` to set for all clusters not specified, and the keyword `interactive` (in place of cluster name) to set for **taskman** jobs. For example:

```
PEAK_INUSE_PERIOD = cluster1 1000 cluster2 700 default 300
```

When defining the interval for LSF Advanced Edition submission clusters, the interval is determined for the entire LSF Advanced Edition mega-cluster (the submission cluster and its execution clusters).

Used for cluster mode only.

When defined in both the Parameters section and the Feature section, the Feature section definition is used for that license feature.

Default

300 seconds

PREEMPT_ORDER

Syntax

```
PREEMPT_ORDER=BY_OWNERSHIP
```

Description

Sets the preemption order based on configured **OWNERSHIP**.

Used for project mode only.

Default

Not defined.

PREEMPT_RESERVE

Syntax

```
PREEMPT_RESERVE=Y | N
```

Description

If `PREEMPT_RESERVE=Y`, enables License Scheduler to preempt either licenses that are reserved or already in use by other projects. The number of jobs must be greater than the number of licenses owned.

If `PREEMPT_RESERVE=N`, License Scheduler does not preempt reserved licenses.

Used for project mode only.

Default

Y. Reserved licenses are preemptable.

RETENTION_FACTOR

Syntax

```
RETENTION_FACTOR=integer%
```

Description

Ensures that when tokens are reclaimed from an overfed cluster, the overfed cluster still gets to dispatch additional jobs, but at a reduced rate. Specify the retention factor as a percentage of tokens to be retained by the overfed cluster.

For example:

```
Begin Feature
NAME = f1
CLUSTER_MODE = Y
CLUSTER_DISTRIBUTION = LanServer(LAN1 1 LAN2 1)
ALLOC_BUFFER = 20
RETENTION_FACTOR = 25%
End Feature
```

With `RETENTION_FACTOR` set, as jobs finish in the overfed cluster and free up tokens, at least 25% of the tokens can be reused by the cluster to dispatch additional jobs. Tokens not held by the cluster are redistributed to other clusters. In general, a higher value means that the process of reclaiming tokens from an overfed cluster takes longer, and an overfed cluster gets to dispatch more jobs while tokens are being reclaimed from it.

When the entire LSF Advanced Edition mega-cluster (the submission cluster and its execution clusters) is overfed, the number of retained tokens is from the entire LSF Advanced Edition mega-cluster.

Used for cluster mode only.

Default

Not defined

SERVICE_DOMAINS

Syntax

`SERVICE_DOMAINS=service_domain_name ...`

service_domain_name

Specify the name of the service domain.

Description

Required if `GROUP_DISTRIBUTION` is defined. Specifies the service domains that provide tokens for this feature.

Only a single service domain can be specified when using cluster mode or fast dispatch project mode.

WORKLOAD_DISTRIBUTION

Syntax

`WORKLOAD_DISTRIBUTION=[service_domain_name(LSF lsf_distribution NON_LSF non_lsf_distribution)] ...`

service_domain_name

Specify a License Scheduler service domain (described in the ServiceDomain section) that distributes the licenses.

lsf_distribution

Specify the share of licenses dedicated to LSF workloads. The share of licenses dedicated to LSF workloads is a ratio of *lsf_distribution:non_lsf_distribution*.

non_lsf_distribution

Specify the share of licenses dedicated to non-LSF workloads. The share of licenses dedicated to non-LSF workloads is a ratio of *non_lsf_distribution:lsf_distribution*.

Description

Defines the distribution given to each LSF and non-LSF workload within the specified service domain.

When running in cluster mode, **WORKLOAD_DISTRIBUTION** can only be specified for WAN service domains; if defined for a LAN feature, it is ignored.

Use **blinfo -a** to display **WORKLOAD_DISTRIBUTION** configuration.

Example

```
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1(Lp1 1 Lp2 2)
WORKLOAD_DISTRIBUTION=LicenseServer1(LSF 8 NON_LSF 2)
End Feature
```

On the `LicenseServer1` domain, the available licenses are dedicated in a ratio of 8:2 for LSF and non-LSF workloads. This means that 80% of the available licenses are dedicated to the LSF workload, and 20% of the available licenses are dedicated to the non-LSF workload.

If `LicenseServer1` has a total of 80 licenses, this configuration indicates that 64 licenses are dedicated to the LSF workload, and 16 licenses are dedicated to the non-LSF workload.

FeatureGroup section

Description

Collects license features into groups. Put `FeatureGroup` sections after `Feature` sections in `lsf.licensescheduler`.

The `FeatureGroup` section is supported in both project mode and cluster mode.

FeatureGroup section structure

The `FeatureGroup` section begins and ends with the lines `Begin FeatureGroup` and `End FeatureGroup`. Feature group definition consists of a unique name and a list of features contained in the feature group.

Example

```
Begin FeatureGroup
NAME = Synposys
FEATURE_LIST = ASTRO VCS_Runtime_Net Hsim Hspice
End FeatureGroup
Begin FeatureGroup
NAME = Cadence
FEATURE_LIST = Encounter NCSim NCVerilog
End FeatureGroup
```

Parameters

- NAME
- FEATURE_LIST

NAME

Syntax

NAME=*feature_group_name*

Required. Defines the name of the feature group. The name must be unique.

FEATURE_LIST

Required. Lists the license features contained in the feature group. The feature names in `FEATURE_LIST` must already be defined in `Feature` sections. Feature names cannot be repeated in the `FEATURE_LIST` of

one feature group. The FEATURE_LIST cannot be empty. Different feature groups can have the same features in their FEATURE_LIST.

ProjectGroup section

Description

Defines the hierarchical relationships of projects.

Used for project mode only. When running in cluster mode, any ProjectGroup sections are ignored.

The hierarchical groups can have multiple levels of grouping. You can configure a tree-like scheduling policy, with the leaves being the license projects that jobs can belong to. Each project group in the tree has a set of values, including shares, limits, ownership and non-shared, or exclusive, licenses.

Use **blstat -G** to view the hierarchical dynamic license information.

Use **blinfo -G** to view the hierarchical configuration.

ProjectGroup section structure

Define a section for each hierarchical group managed by License Scheduler.

The keywords GROUP, SHARES, OWNERSHIP, LIMIT, and NON_SHARED are required. The keywords PRIORITY and DESCRIPTION are optional. Empty brackets are allowed only for OWNERSHIP, LIMIT, and PRIORITY. SHARES must be specified.

```
Begin ProjectGroup
GROUP      SHARES      OWNERSHIP LIMITS  NON_SHARED PRIORITY
(root(A B C)) (1 1 1)      ()          ()      ()      (3 2 -)
(A (P1 D))  (1 1)        ()          ()      ()      (3 5)
(B (P4 P5)) (1 1)        ()          ()      ()      ()
(C (P6 P7 P8)) (1 1 1)     ()          ()      ()      (8 3 0)
(D (P2 P3)) (1 1)        ()          ()      ()      (2 1)
End ProjectGroup
```

If desired, ProjectGroup sections can be completely independent, without any overlapping projects.

```
Begin ProjectGroup
GROUP      SHARES      OWNERSHIP LIMITS  NON_SHARED
(digital_sim (sim sim_reg)) (40 60) (100 0) () ()
End ProjectGroup

Begin ProjectGroup
GROUP      SHARES      OWNERSHIP LIMITS  NON_SHARED
(analog_sim (app1 multitoken app1_reg)) (50 10 40) (65 25 0) (- 50 -) ()
End digital
```

Parameters

- DESCRIPTION
- GROUP
- LIMITS
- NON_SHARED
- OWNERSHIP
- PRIORITY
- SHARES

DESCRIPTION

Description of the project group.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\). The maximum length for the text is 64 characters. When the DESCRIPTION column is not empty it should contain one entry for each project group member.

For example:

GROUP	SHARES	OWNERSHIP	LIMITS	NON_SHARED	DESCRIPTION
(R (A B))	(1 1)	()	()	(10 10)	()
(A (p1 p2))	(1 1)	(40 60)	()	()	("p1 desc." " ")
(B (p3 p4))	(1 1)	()	()	()	("p3 desc." "p4 desc.")

Use **blinfo -G** to view hierarchical project group descriptions.

GROUP

Defines the project names in the hierarchical grouping and its relationships. Each entry specifies the name of the hierarchical group and its members.

For better readability, specify the projects in the order from the root to the leaves.

Specify the entry as follows:

(group (member ...))

LIMITS

Defines the maximum number of licenses that can be used at any one time by the hierarchical group member projects. Specify the maximum number of licenses for each member, separated by spaces, in the same order as listed in the GROUP column.

A dash (-) is equivalent to INFINIT_INT, which means there is no maximum limit and the project group can use as many licenses as possible.

You can leave the parentheses empty () if desired.

NON_SHARED

Defines the number of licenses that the hierarchical group member projects use exclusively. Specify the number of licenses for each group or project, separated by spaces, in the same order as listed in the GROUP column.

A dash (-) is equivalent to a zero, which means there are no licenses that the hierarchical group member projects use exclusively.

For hierarchical project groups in fast dispatch project mode, LSF License Scheduler ignores the NON_SHARED value configured for project groups, and only uses the NON_SHARED value for the child projects. The project group's NON_SHARED value is the sum of the NON_SHARED values of its child projects.

Normally, the total number of non-shared licenses should be less than the total number of license tokens available. License tokens may not be available to project groups if the total non-shared licenses for all groups is greater than the number of shared tokens available.

For example, feature p4_4 is configured as follows, with a total of 4 tokens:

```
Begin Feature
NAME =p4_4 # total token value is 4
GROUP_DISTRIBUTION=final
SERVICE_DOMAINS=LanServer
End Feature
```

The correct configuration is:

GROUP	SHARES	OWNERSHIP	LIMITS	NON_SHARED
(final (G2 G1))	(1 1)	()	()	(2 0)
(G1 (AP2 AP1))	(1 1)	()	()	(1 1)

Valid values

Any positive integer up to the LIMITS value defined for the specified hierarchical group.

If defined as greater than LIMITS, NON_SHARED is set to LIMITS.

OWNERSHIP

Defines the level of ownership of the hierarchical group member projects. Specify the ownership for each member, separated by spaces, in the same order as listed in the GROUP column.

You can only define OWNERSHIP for hierarchical group member projects, not hierarchical groups. Do not define OWNERSHIP for the top level (root) project group. Ownership of a given internal node is the sum of the ownership of all child nodes it directly governs.

A dash (-) is equivalent to a zero, which means there are no owners of the projects. You can leave the parentheses empty () if desired.

Valid values

A positive integer between the NON_SHARED and LIMITS values defined for the specified hierarchical group.

- If defined as less than NON_SHARED, OWNERSHIP is set to NON_SHARED.
- If defined as greater than LIMITS, OWNERSHIP is set to LIMITS.

PRIORITY

Defines the priority assigned to the hierarchical group member projects. Specify the priority for each member, separated by spaces, in the same order as listed in the GROUP column.

“0” is the lowest priority, and a higher number specifies a higher priority. This column overrides the default behavior. Instead of preempting based on the accumulated inuse usage of each project, the projects are preempted according to the specified priority from lowest to highest.

By default, priorities are evaluated top down in the project group hierarchy. The priority of a given node is first decided by the priority of the parent groups. When two nodes have the same priority, priority is determined by the accumulated inuse usage of each project at the time the priorities are evaluated. Specify LS_PREEMPT_PEER=Y in the Parameters section to enable bottom-up license token preemption in hierarchical project group configuration.

A dash (-) is equivalent to a zero, which means there is no priority for the project. You can leave the parentheses empty () if desired.

Use **blinfo -G** to view hierarchical project group priority information.

Priority of default project

If not explicitly configured, the default project has the priority of 0. You can override this value by explicitly configuring the default project in Projects section with the chosen priority value.

SHARES

Required. Defines the shares assigned to the hierarchical group member projects. Specify the share for each member, separated by spaces, in the same order as listed in the GROUP column.

Projects section**Description**

Required for project mode only. Ignored in cluster mode. Lists the License Scheduler projects.

Projects section structure

The Projects section begins and ends with the lines `Begin Projects` and `End Projects`. The second line consists of the required column heading `PROJECTS` and the optional column heading `PRIORITY`. Subsequent lines list participating projects, one name per line.

Examples

The following example lists the projects without defining the priority:

```
Begin Projects
PROJECTS
Lp1
Lp2
Lp3
Lp4
...
End Projects
```

The following example lists the projects and defines the priority of each project:

```
Begin Projects
PROJECTS      PRIORITY
Lp1           3
Lp2           4
Lp3           2
Lp4           1
default       0
...
End Projects
```

Parameters

- DESCRIPTION
- PRIORITY
- PROJECTS

DESCRIPTION

Description of the project.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (`\`). The maximum length for the text is 64 characters.

Use **blinfo -Lp** to view the project description.

PRIORITY

Defines the priority for each project where “0” is the lowest priority, and the higher number specifies a higher priority. This column overrides the default behavior. Instead of preempting in order the projects are listed under `PROJECTS` based on the accumulated inuse usage of each project, the projects are preempted according to the specified priority from lowest to highest.

Used for project mode only.

When 2 projects have the same priority number configured, the first project listed has higher priority, like LSF queues.

Use **blinfo -Lp** to view project priority information.

Priority of default project

If not explicitly configured, the default project has the priority of 0. You can override this value by explicitly configuring the default project in Projects section with the chosen priority value.

PROJECTS

Defines the name of each participating project. Specify using one name per line.

Automatic time-based configuration

Variable configuration is used to automatically change License Scheduler license token distribution policy configuration based on time windows. You define automatic configuration changes in `lsf.licensescheduler` by using if-else constructs and time expressions in the Feature section. After you change the file, check the configuration with the **bladmin ckconfig** command, and restart License Scheduler in the cluster with the **bladmin reconfig** command.

Used for both project mode and cluster mode.

The expressions are evaluated by License Scheduler every 10 minutes based on the **bld** start time. When an expression evaluates true, License Scheduler dynamically changes the configuration based on the associated configuration statements, restarting **bld** automatically.

When LSF determines a feature has been added, removed, or changed, `mbatchd` no longer restarts automatically. Instead a message indicates that a change has been detected, prompting the user to restart manually with **badmin mbdrestart**.

This affects automatic time-based configuration in the Feature section of `lsf.licensescheduler`. When **mbatchd** detects a change in the Feature configuration, you must restart **mbatchd** for the change to take effect.

Example

```
Begin Feature
NAME = f1
#if time(5:16:30-1:8:30 20:00-8:30)
DISTRIBUTION=Lan(P1 2/5 P2 1)
#elif time(3:8:30-3:18:30)
DISTRIBUTION=Lan(P3 1)
#else
DISTRIBUTION=Lan(P1 1 P2 2/5)
#endif
End Feature
```

lsf.shared

The `lsf.shared` file contains common definitions that are shared by all load sharing clusters defined by `lsf.cluster.cluster_name` files.

This includes lists of cluster names, host types, host models, the special resources available, and external load indices, including indices required to submit jobs using JSDL files.

This file is installed by default in the directory defined by `LSF_CONFDIR`.

Changing lsf.shared configuration

After making any changes to `lsf.shared`, run the following commands:

- **lsadmin reconfig** to reconfigure LIM
- **badmin mbdrestart** to restart `mbatchd`

#INCLUDE

Syntax

```
#INCLUDE "path-to-file"
```

Description

Inserts a configuration setting from another file to the current location. Use this directive to dedicate control of a portion of the configuration to other users or user groups by providing write access for the included file to specific users or user groups, and to ensure consistency of configuration file settings in different clusters (if you are using the LSF multicluster capability).

See more information on shared configuration file content in *Administering IBM Spectrum LSF*

All `#INCLUDE` lines must be inserted at the beginning of the local configuration file. If placed within or after any other sections, LSF reports an error.

Cluster section

(Required) Lists the cluster names recognized by the LSF system

Cluster section structure

The first line must contain the mandatory keyword `ClusterName`. The other keyword is optional.

The first line must contain the mandatory keyword `ClusterName` and the keyword `Servers` when using the LSF multicluster capability.

Each subsequent line defines one cluster.

Example Cluster section

```
Begin Cluster
ClusterName Servers
cluster1      (hostA hostB)
cluster2      (hostD)
End Cluster
```

ClusterName

Defines all cluster names recognized by the LSF system.

All cluster names referenced anywhere in the LSF system must be defined here. The file names of cluster-specific configuration files must end with the associated cluster name.

By default, if MultiCluster is installed, all clusters listed in this section participate in the same MultiCluster environment. However, individual clusters can restrict their MultiCluster participation by specifying a subset of clusters at the cluster level (`lsf.cluster.cluster_name RemoteClusters` section).

Servers

MultiCluster only. List of hosts in this cluster that LIMs in remote clusters can connect to and obtain information from.

For other clusters to work with this cluster, one of these hosts must be running `mbatchd`.

HostType section

(Required) Lists the valid host types in the cluster. All hosts that can run the same binary executable are in the same host type.



CAUTION:

If you remove `NTX86`, `NTX64`, or `NTIA64` from the HostType section, the functionality of `lspasswd.exe` is affected. The `lspasswd` command registers a password for a Windows user account.

HostType section structure

The first line consists of the mandatory keyword `TYPENAME`.

Subsequent lines name valid host types.

Example HostType section

```
Begin HostType
TYPENAME
SOL64
SOLSPARC
LINUX86LINUXPPC
LINUX64
NTX86
NTX64
NTIA64
End HostType
```

TYPENAME

Host type names are usually based on a combination of the hardware name and operating system. If your site already has a system for naming host types, you can use the same names for LSF.

HostModel section

(Required) Lists models of machines and gives the relative CPU scaling factor for each model. All hosts of the same relative speed are assigned the same host model.

LSF uses the relative CPU scaling factor to normalize the CPU load indices so that jobs are more likely to be sent to faster hosts. The CPU factor affects the calculation of job execution time limits and accounting. Using large or inaccurate values for the CPU factor can cause confusing results when CPU time limits or accounting are used.

HostModel section structure

The first line consists of the mandatory keywords MODELNAME, CPUFACTOR, and ARCHITECTURE.

Subsequent lines define a model and its CPU factor.

Example HostModel section

```
Begin HostModel MODELNAME CPUFACTOR ARCHITECTURE
PC400 13.0 (i86pc_400 i686_400)
PC450 13.2 (i86pc_450 i686_450)
Sparc5F 3.0 (SUNWSPARCstation5_170_sparc)
Sparc20 4.7 (SUNWSPARCstation20_151_sparc)
Ultra5S 10.3 (SUNWUltra5_270_sparcv9 SUNWUltra510_270_sparcv9)
End HostModel
```

ARCHITECTURE

(Reserved for system use only) Indicates automatically detected host models that correspond to the model names.

CPUFACTOR

Though it is not required, you would typically assign a CPU factor of 1.0 to the slowest machine model in your system and higher numbers for the others. For example, for a machine model that executes at twice the speed of your slowest model, a factor of 2.0 should be assigned.

MODELNAME

Generally, you need to identify the distinct host types in your system, such as MIPS and SPARC first, and then the machine models within each, such as SparcIPC, Sparc1, Sparc2, and Sparc10.

About automatically detected host models and types

When you first install LSF, you do not necessarily need to assign models and types to hosts in `lsf.cluster.cluster_name`. If you do not assign models and types to hosts in `lsf.cluster.cluster_name`, LIM automatically detects the model and type for the host.

If you have versions earlier than LSF 4.0, you may have host models and types already assigned to hosts. You can take advantage of automatic detection of host model and type also.

Automatic detection of host model and type is useful because you no longer need to make changes in the configuration files when you upgrade the operating system or hardware of a host and reconfigure the cluster. LSF will automatically detect the change.

Mapping to CPU factors

Automatically detected models are mapped to the short model names in `lsf.shared` in the ARCHITECTURE column. Model strings in the ARCHITECTURE column are only used for mapping to the short model names.

Example `lsf.shared` file:

```
Begin HostModel
MODELNAME  CPUFACTOR  ARCHITECTURE
SparcU5    5.0           (SUNWUltra510_270_sparcv9)
PC486     2.0           (i486_33 i486_66)
PowerPC   3.0           (PowerPC12 PowerPC16 PowerPC31)
End HostModel
```

If an automatically detected host model cannot be matched with the short model name, it is matched to the best partial match and a warning message is generated.

If a host model cannot be detected or is not supported, it is assigned the DEFAULT model name and an error message is generated.

Naming convention

Models that are automatically detected are named according to the following convention:

```
hardware_platform [_processor_speed [_processor_type]]
```

where:

- *hardware_platform* is the only mandatory component
- *processor_speed* is the optional clock speed and is used to differentiate computers within a single platform
- *processor_type* is the optional processor manufacturer used to differentiate processors with the same speed
- Underscores (`_`) between *hardware_platform*, *processor_speed*, *processor_type* are mandatory.

Resource section

Optional. Defines resources (must be done by the LSF administrator).

Resource section structure

The first line consists of the keywords. RESOURCENAME and DESCRIPTION are mandatory. The other keywords are optional. Subsequent lines define resources.

Example Resource section

```
Begin Resource
RESOURCENAME  TYPE      INTERVAL  INCREASING  CONSUMABLE  DESCRIPTION  # Keywords
patchrev     Numeric  ()        Y           ()          (Patch revision)
specman      Numeric  ()        N           ()          (Specman)
```

```

switch      Numeric  ()      Y      N      (Network Switch)
rack        String   ()      ()      (Server room rack)
owner       String   ()      ()      (Owner of the host)
elimres     Numeric  10     Y      ()      (elim generated index)
ostype      String   ()      ()      (Operating system and version)
lmhostid    String   ()      ()      (FlexLM's lmhostid)
limversion  String   ()      ()      (Version of LIM binary)
End Resource

```

RESOURCENAME

The name you assign to the new resource. An arbitrary character string.

- A resource name cannot begin with a number.
- A resource name cannot contain any of the following special characters:

```
: . ( ) [ + - * / ! & | < > @ = ,
```

- A resource name cannot be any of the following reserved names:

```
cpu cpuf io logins ls idle maxmem maxswp maxtmp type model status it
mem ncpus define_ncpus_cores define_ncpus_procs
define_ncpus_threads ndisks pg r15m r15s r1m swap swp tmp ut
```

- To avoid conflict with `inf` and `nan` keywords in 3rd-party libraries, resource names should not begin with `inf` or `nan` (upper case or lower case). Resource requirement strings, such as `-R "infra"` or `-R "nano"` will cause an error. Use `-R "defined(infixx)"` or `-R "defined(nanxx)"`, to specify these resource names.
- Resource names are case sensitive
- Resource names can be up to 39 characters in length
- For Solaris machines, the keyword `int` is reserved and cannot be used.

TYPE

The type of resource:

- Boolean—Resources that have a value of 1 on hosts that have the resource and 0 otherwise.
- Numeric—Resources that take numerical values, such as all the load indices, number of processors on a host, or host CPU factor.
- String—Resources that take string values, such as host type, host model, host status.

Default

If TYPE is not given, the default type is Boolean.

INTERVAL

Optional. Applies to dynamic resources only.

Defines the time interval (in seconds) at which the resource is sampled by the ELIM.

If INTERVAL is defined for a numeric resource, it becomes an external load index.

Default

If INTERVAL is not given, the resource is considered static.

INCREASING

Applies to numeric resources only.

If a larger value means greater load, INCREASING should be defined as Y. If a smaller value means greater load, INCREASING should be defined as N.

CONSUMABLE

Explicitly control if a resource is consumable. Applies to static or dynamic numeric resources.

Static and dynamic numeric resources can be specified as consumable. CONSUMABLE is optional. The defaults for the consumable attribute are:

- Built-in indicies:
 - The following are consumable: r15s, r1m, r15m, ut, pg, io, ls, it, tmp, swp, mem.
 - All other built-in static resources are not consumable. (e.g., ncpus, ndisks, maxmem, maxswp, maxtmp, cpuf, type, model, status, rexpri, server, hname).
- External shared resources:
 - All numeric resources are consumable.
 - String and boolean resources are not consumable.

You should only specify consumable resources in the rusage section of a resource requirement string. Non-consumable resources are ignored in rusage sections.

A non-consumable resource should not be releasable. Non-consumable numeric resource should be able to used in order, select and same sections of a resource requirement string.

When LSF_STRICT_RESREQ=Y in `lsf.conf`, LSF rejects resource requirement strings where an rusage section contains a non-consumable resource.

DESCRIPTION

Brief description of the resource.

The information defined here will be returned by the `ls_info()` API call or printed out by the **lsinfo** command as an explanation of the meaning of the resource.

RELEASE

Applies to numeric shared resources only.

Controls whether LSF releases the resource when a job using the resource is suspended. When a job using a shared resource is suspended, the resource is held or released by the job depending on the configuration of this parameter.

Specify N to hold the resource, or specify Y to release the resource.

Default

N

lsf.sudoers

About lsf.sudoers

The `lsf.sudoers` file is an optional file to configure security mechanisms. It is not installed by default.

You use `lsf.sudoers` to set the parameter `LSF_EAUTH_KEY` to configure a key for `eauth` to encrypt and decrypt user authentication data.

On UNIX, you also use `lsf.sudoers` to grant permission to users other than root to perform certain operations as root in LSF, or as a specified user.

These operations include:

- LSF daemon startup/shutdown
- User ID for LSF authentication
- User ID for LSF pre- and post-execution commands.

- User ID for external LSF executables

If `lsf.sudoers` does not exist, only root can perform these operations in LSF on UNIX.

On UNIX, this file is located in `/etc`.

There is one `lsf.sudoers` file per host.

On Windows, this file is located in the directory specified by the **LSF_SECUREDIR** parameter in the `lsf.conf` file.

Changing lsf.sudoers configuration

After making any changes to `lsf.sudoers`, run **badadmin reconfig** to reload the configuration files.

lsf.sudoers on UNIX

In LSF, certain operations such as daemon startup can only be performed by root. The `lsf.sudoers` file grants root privileges to specific users or user groups to perform these operations.

Location

`lsf.sudoers` must be located in `/etc` on each host.

Permissions

`lsf.sudoers` must have permission 600 and be readable and writable only by root.

lsf.sudoers on Windows

The `lsf.sudoers` file is shared over an NTFS network, not duplicated on every Windows host.

By default, LSF installs `lsf.sudoers` in the `%SYSTEMROOT%` directory.

The location of `lsf.sudoers` on Windows must be specified by `LSF_SECUREDIR` in `lsf.conf`. You must configure the `LSF_SECUREDIR` parameter in `lsf.conf` if using `lsf.sudoers` on Windows.

Windows permissions

Restriction:

The owner of `lsf.sudoers` on Windows be Administrators. If not, eauth may not work.

The permissions on `lsf.sudoers` for Windows are:

Workgroup Environment

- Local Admins (W)
- Everyone (R)

Domain Environment

- Domain Admins (W)
- Everyone (R)

File format

The format of `lsf.sudoers` is very similar to that of `lsf.conf`.

Each entry can have one of the following forms:

- NAME=VALUE
- NAME=
- NAME= "STRING1 STRING2 ..."

The equal sign = must follow each NAME even if no value follows and there should be no space beside the equal sign.

NAME describes an authorized operation.

VALUE is a single string or multiple strings separated by spaces and enclosed in quotation marks.

Lines starting with a pound sign (#) are comments and are ignored. Do not use `#if` as this is reserved syntax for time-based configuration.

Example lsf.sudoers File

```
LSB_PRE_POST_EXEC_USER=user100
LSF_STARTUP_PATH=/usr/share/lsf/etc
LSF_STARTUP_USERS="user1 user10 user55"
```

Creating and modifying lsf.sudoers

You can create and modify `lsf.sudoers` with a text editor.

After you modify `lsf.sudoers`, you must run **admin hrestart all** to restart all sbatchds in the cluster with the updated configuration.

Parameters

- LSB_PRE_POST_EXEC_USER
- LSF_EAUTH_KEY
- LSF_EAUTH_USER
- LSF_EEXEC_USER
- LSF_EGO_ADMIN_PASSWD
- LSF_EGO_ADMIN_USER
- LSF_LOAD_PLUGINS
- LSF_STARTUP_PATH
- LSF_STARTUP_USERS

LSB_PRE_POST_EXEC_USER

Syntax

```
LSB_PRE_POST_EXEC_USER=user_name
```

Description

Specifies the UNIX user account under which pre- and post-execution commands run. This parameter affects host-based pre- and post-execution processing defined at the first level.

You can specify only one user account. If the pre-execution or post-execution commands perform privileged operations that require root permissions on UNIX hosts, specify a value of `root`.

If you configure this parameter as `root`, the **LD_PRELOAD** and **LD_LIBRARY_PATH** variables are removed from the pre-execution, post-execution, and `eexec` environments for security purposes.

Default

Not defined. Pre-execution and post-execution commands run under the user account of the user who submits the job.

LSF_EAUTH_KEY

Syntax

```
LSF_EAUTH_KEY=key
```

Description

Applies to UNIX, Windows, and mixed UNIX/Windows clusters.

Specifies the key that **eauth** uses to encrypt and decrypt user authentication data. Defining this parameter enables increased security at your site. The key must contain at least six characters and must use only printable characters.

For UNIX, you must edit the `lsf.sudoers` file on all hosts within the cluster and specify the same encryption key. For Windows, you must edit the shared `lsf.sudoers` file.

Default

Not defined. The `eauth` executable encrypts and decrypts authentication data using an internal key.

LSF_EAUTH_USER**Syntax**

`LSF_EAUTH_USER=user_name`

Description

UNIX only.

Specifies the UNIX user account under which the external authentication executable **eauth** runs.

Default

Not defined. The `eauth` executable runs under the account of the primary LSF administrator.

LSF_EEXEC_USER**Syntax**

`LSF_EEXEC_USER=user_name`

Description

UNIX only.

Specifies the UNIX user account under which the external executable `eexec` runs.

Default

Not defined. The `eexec` executable runs under root or the account of the user who submitted the job.

LSF_EGO_ADMIN_PASSWD**Syntax**

`LSF_EGO_ADMIN_PASSWD=password`

Description

When the EGO Service Controller (EGOSC) is configured to control LSF daemons, enables UNIX and Windows users to bypass the additional login required to start `res` and `sbatchd`. Bypassing the EGO administrator login enables the use of scripts to automate system startup.

Specify the Admin EGO cluster administrator password as clear text. You must also define the **LSF_EGO_ADMIN_USER** parameter.

Default

Not defined. With EGOSC daemon control enabled, the **lsadmin** and **badmin** startup subcommands invoke the **egosh user logon** command to prompt for the Admin EGO cluster administrator credentials.

LSF_EGO_ADMIN_USER

Syntax

LSF_EGO_ADMIN_USER=Admin

Description

When the EGO Service Controller (EGOSC) is configured to control LSF daemons, enables UNIX and Windows users to bypass the additional login required to start **res** and **sbatchd**. Bypassing the EGO administrator login enables the use of scripts to automate system startup.

Specify the Admin EGO cluster administrator account. You must also define the **LSF_EGO_ADMIN_PASSWD** parameter.

Default

Not defined. With EGOSC daemon control enabled, the **lsadmin** and **badmin** startup subcommands invoke the **egosh user logon** command to prompt for the Admin EGO cluster administrator credentials.

LSF_LOAD_PLUGINS

Syntax

LSF_LOAD_PLUGINS=y | Y

Description

If defined, LSF loads plugins from **LSB_LSBDIR**. Used for Kerberos authentication and to enable the LSF cpuset plugin for SGI.

Default

Not defined. LSF does not load plugins.

LSF_STARTUP_PATH

Syntax

LSF_STARTUP_PATH=*path*

Description

UNIX only. Enables the LSF daemon startup control feature when **LSF_STARTUP_USERS** is also defined. Define both parameters when you want to allow users other than root to start LSF daemons.

Specifies the absolute path name of the directory in which the LSF daemon binary files (**lim**, **res**, **sbatchd**, and **mbatchd**) are installed. LSF daemons are usually installed in the path specified by **LSF_SERVERDIR** defined in the **cshrc.lsf**, **profile.lsf** or **lsf.conf** files.

Important:

For security reasons, you should move the LSF daemon binary files to a directory other than **LSF_SERVERDIR** or **LSF_BINDIR**. The user accounts specified by **LSF_STARTUP_USERS** can start any binary in the **LSF_STARTUP_PATH**.

Default

Not defined. Only the root user account can start LSF daemons.

LSF_STARTUP_USERS**Syntax**

```
LSF_STARTUP_USERS=all_admins | "user_name..."
```

Description

UNIX only. Enables the LSF daemon startup control feature when **LSF_STARTUP_PATH** is also defined. Define both parameters when you want to allow users other than root to start LSF daemons. On Windows, the services admin group is equivalent to **LSF_STARTUP_USERS**.

On UNIX hosts, by default only root can start LSF daemons. To manually start LSF daemons, a user runs the commands **lsadmin** and **badmin**, which have been installed as setuid root. **LSF_STARTUP_USERS** specifies a list of user accounts that can successfully run the commands **lsadmin** and **badmin** to start LSF daemons.

all_admins

- Allows all UNIX users defined as LSF administrators in the file `lsf.cluster.cluster_name` to start LSF daemons as root by running the **lsadmin** and **badmin** commands.
- Not recommended due to the security risk of a non-root LSF administrator adding to the list of administrators in the `lsf.cluster.cluster_name` file.
- Not required for Windows hosts because all users with membership in the services admin group can start LSF daemons.

"user_name..."

- Allows the specified user accounts to start LSF daemons by running the **lsadmin** and **badmin** commands.
- Separate multiple user names with a space.
- For a single user, do not use quotation marks.

Default

Not defined. Only the root user account can start LSF daemons.

See also

LSF_STARTUP_PATH

lsf.task

Users should not have to specify a resource requirement each time they submit a job. LSF supports the concept of a task list. This chapter describes the files used to configure task lists: `lsf.task`, `lsf.task.cluster_name`, and `.lsftask`.

Changing task list configuration

After making any changes to the task list files, run the following commands:

- **lsadmin reconfig** to reconfigure LIM
- **badmin reconfig** to reload the configuration files

About task lists

A task list is a list in LSF that keeps track of the default resource requirements for different applications and task eligibility for remote execution.

The term task refers to an application name. With a task list defined, LSF automatically supplies the resource requirement of the job whenever users submit a job unless one is explicitly specified at job submission.

LSF takes the job's command name as the task name and uses that name to find the matching resource requirement for the job from the task list. If a task does not have an entry in the task list, LSF assumes the default resource requirement; that is, a host that has the same host type as the submission host will be chosen to run the job.

An application listed in a task file is considered for load sharing by its placement in either the local tasks or remote tasks list.

- A local task is typically an application or command that it does not make sense to run remotely such as **ls**.
- A remote task is an application or command that can be run on another machine in the LSF cluster. The **compress** command is an example of a remote task.

Some applications require resources other than the default. LSF can store resource requirements for specific applications in remote task list files, so that LSF automatically chooses candidate hosts that have the correct resources available.

For frequently used commands and software packages, the LSF administrator can set up cluster-wide resource requirements that apply to all users in the cluster.

Users can modify and add to these requirements by setting up additional resource requirements that apply only to their own jobs.

Cluster-wide resource requirements

The resource requirements of applications are stored in the remote task list file.

LSF automatically picks up a job's default resource requirement string from the remote task list files, unless you explicitly override the default by specifying the resource requirement string on the command line.

User-level resource requirements

You may have applications that you need to control yourself. Perhaps your administrator did not set them up for load sharing for all users, or you need a non-standard setup. You can use LSF commands to find out resource names available in your system, and tell LSF about the needs of your applications. LSF stores the resource requirements for you from then on.

You can specify resource requirements when tasks are added to the user's remote task list. If the task to be added is already in the list, its resource requirements are replaced.

```
lsrtasks + myjob/swap>=100 && cpu
```

This adds myjob to the remote tasks list with its resource requirements.

Task files

There are 3 task list files that can affect a job:

- `lsf.task` - system-wide defaults apply to all LSF users, even across multiple clusters if MultiCluster is installed
- `lsf.task.cluster_name` - cluster-wide defaults apply to all users in the cluster
- `$HOME/.lsftask` - user-level defaults apply to a single user. This file lists applications to be added to or removed from the default system lists for your jobs. Resource requirements specified in this file override those in the system lists.

The clusterwide task file is used to augment the systemwide file. The user's task file is used to augment the systemwide and clusterwide task files.

LSF combines the systemwide, clusterwide, and user-specific task lists for each user's view of the task list. In cases of conflicts, such as different resource requirements specified for the same task in different lists, the clusterwide list overrides the systemwide list, and the user-specific list overrides both.

LSF_CONFDIR/lsf.task

Systemwide task list applies to all clusters and all users.

This file is used in a MultiCluster environment.

LSF_CONFDIR/lsf.task.cluster_name

Clusterwide task list applies to all users in the same cluster.

\$HOME/.lsftask

User task list, one per user, applies only to the specific user. This file is automatically created in the user's home directory whenever a user first updates his task lists using the **lsrtasks** or **lsltasks** commands. For details about task eligibility lists, see the `ls_task(3)` API reference man page.

Permissions

Only the LSF administrator can modify the systemwide task list (`lsf.task`) and the clusterwide task list (`lsf.task.cluster_name`).

A user can modify his own task list (`.lsftask`) with the **lsrtasks** and **lsltasks** commands.

Format of task files

Each file consists of two sections, LocalTasks and RemoteTasks. For example:

```
Begin LocalTasks
ps
hostname
uname
crontab
End LocalTasks
Begin RemoteTasks
+ "newjob/mem>25"
+ "verilog/select[type==any && swp>100]"
make/cpu
nroff/-
End RemoteTasks
```

Tasks are listed one per line. Each line in a section consists of a task name, and, for the RemoteTasks section, an optional resource requirement string separated by a slash (/).

setup.config

A plus sign (+) or a minus sign (-) can optionally precede each entry. If no + or - is specified, + is assumed.

A + before a task name means adding a new entry (if non-existent) or replacing an entry (if already existent) in the task list. A - before a task name means removing an entry from the application's task lists if it was already created by reading higher level task files.

LocalTasks section

The section starts with `Begin LocalTasks` and ends with `End LocalTasks`.

This section lists tasks that are not eligible for remote execution, either because they are trivial tasks or because they need resources on the local host.

RemoteTasks section

The section starts with `Begin RemoteTasks` and ends with `End RemoteTasks`.

This section lists tasks that are eligible for remote execution. You can associate resource requirements with each task name.

See *Administering IBM Spectrum LSF* for information about resource requirement strings. If the resource requirement string is not specified for a remote task, the default is `"select[type==local] order[r15s:pg]"`.

setup.config

About setup.config

The `setup.config` file contains options for License Scheduler installation and configuration for systems without LSF. You only need to edit this file if you are installing License Scheduler as a standalone product without LSF.

Template location

A template `setup.config` is included in the License Scheduler installation script tar file and is located in the directory created when you uncompress and extract the installation script tar file. Edit the file and uncomment the options you want in the template file. Replace the example values with your own settings to specify the options for your new License Scheduler installation.

Important: The sample values in the `setup.config` template file are examples only. They are not default installation values.

After the License Scheduler installation, the `setup.config` containing the options you specified is located in `LS_TOP/10.1/install/`.

Format

Each entry in `setup.config` has the form:

```
NAME="STRING1 STRING2 ..."
```

The equal sign = must follow each NAME even if no value follows and there should be no spaces around the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Blank lines and lines starting with a pound sign (#) are ignored.

Parameters

- LS_ADMIN

- LS_HOSTS
- LS_LMSTAT_PATH
- LS_TOP

LS_ADMIN

Syntax

```
LS_ADMIN="user_name [user_name ... ]"
```

Description

Lists the License Scheduler administrators. The first user account name in the list is the primary License Scheduler administrator.

The primary License Scheduler administrator account is typically named lsadmin.



CAUTION: You should *not* configure the root account as the primary License Scheduler administrator.

Valid Values

User accounts for License Scheduler administrators must exist on all hosts using License Scheduler prior to installation.

Example

```
LS_ADMIN="lsadmin user1 user2"
```

Default

The user running the License Scheduler installation script.

LS_HOSTS

Syntax

```
LS_HOSTS="host_name [host_name ... ]"
```

Description

Defines a list of hosts that are candidates to become License Scheduler master hosts. Provide at least one host from which the License Scheduler daemon will run.

Valid Values

Any valid License Scheduler host name.

Example

```
LS_HOSTS="host_name1 host_name2"
```

Default

The local host in which the License Scheduler installation script is running.

LS_LMSTAT_PATH

Syntax

```
LS_LMSTAT_PATH="/path"
```

Description

Defines the full path to the **lmstat** program. License Scheduler uses **lmstat** to gather the FlexNet license information for scheduling. This path does not include the name of the **lmstat** program itself.

Example

```
LS_LMSTAT_PATH="/usr/bin"
```

Default

The installation script attempts to find a working copy of **lmstat** on the current system. If it is unsuccessful, the path is set as blank ("").

LS_TOP

Syntax

```
LS_TOP="/path"
```

Description

Defines the full path to the top level License Scheduler installation directory.

Valid Values

Must be an absolute path to a shared directory that is accessible to all hosts using License Scheduler. Cannot be the root directory (/).

Recommended Value

The file system containing LS_TOP must have enough disk space for all host types (approximately 1.5 GB per host type).

Example

```
LS_TOP="/usr/share/lis"
```

Default

None — required variable

slave.config

The `slave.config` file contains options for installing and configuring a server host that can be dynamically added or removed.

Dynamically added LSF hosts that are not master candidates are *server hosts*. Each dynamic server host has its own LSF binary files, local `lsf.conf` file, and shell environment script files (`cshrc.lsf` and `profile.lsf`). You must install LSF on each server host.

Use the **lsfinstall -s -f slave.config** command to install LSF with the options that are specified in the `slave.config` file.

Template location

A template `slave.config` file is located in the installation script directory that is created when you extract the installer script package. Edit the file and uncomment the options that you want in the template file. Replace the example values with your own settings to specify the options for your new LSF installation.

Important:

The sample values in the `slave.config` template file are examples only. They are not default installation values.

Format

Each entry in the `slave.config` file has the form:

```
NAME="STRING1 STRING2 ..."
```

Do not use spaces around the equal sign `=`, which must follow each NAME even if no value follows.

A value that contains multiple strings that are separated by spaces must be enclosed in quotation marks.

Blank lines and lines that begin with a number sign (`#`) are ignored.

Parameters

- EGO_DAEMON_CONTROL
- ENABLE_EGO
- EP_BACKUP
- LSF_ADMINS
- LSF_ENTITLEMENT_FILE
- LSF_LIM_PORT
- LSF_SERVER_HOSTS
- LSF_TARDIR
- LSF_LOCAL_RESOURCES
- LSF_TOP
- SILENT_INSTALL
- LSF_SILENT_INSTALL_TARLIST

EGO_DAEMON_CONTROL

Enables EGO to control the LSF `res` and `sbatchd` daemons.

Syntax

```
EGO_DAEMON_CONTROL="Y" | "N"
```

Description

Set the value to "Y" if you want the EGO service controller to start the **res** and **sbatchd** daemons, and restart if they fail.

All hosts in the cluster must use the same value for this parameter. Therefore, the value of the **EGO_DAEMON_CONTROL** parameter in the `slave.config` file must be the same as the specification for the **EGO_DAEMON_CONTROL** parameter in the `install.config` file).

slave.config

To avoid conflicts, leave this parameter undefined if you use a script to start the LSF daemons.

Note:

If you specify **ENABLE_EGO="N"**, this parameter is ignored.

Example

```
EGO_DAEMON_CONTROL="N"
```

Default

N (the **res** and **sbatchd** daemons are started manually)

ENABLE_EGO

Enables EGO functions in the LSF cluster.

Syntax

```
ENABLE_EGO="Y" | "N"
```

Description

ENABLE_EGO="Y"

Causes the **lsfinstall** command to uncomment the **LSF_EGO_ENVDIR** parameter and sets the **LSF_ENABLE_EGO** parameter to "Y" in the `lsf.conf` file.

ENABLE_EGO="N"

Causes the **lsfinstall** command to comment out the **LSF_EGO_ENVDIR** parameter and sets the **LSF_ENABLE_EGO** parameter to "N" in the `lsf.conf` file.

Set the value to "Y" to take advantage of the following LSF features that depend on EGO:

- LSF daemon control by the EGO service controller
- EGO enabled SLA scheduling

Default

N (EGO is disabled in the LSF cluster)

EP_BACKUP

Enables backup and rollback for enhancement packs.

Syntax

```
EP_BACKUP="Y" | "N"
```

Description

Set the value to "N" to disable backups when you install enhancement packs. You will not be able to roll back to the previous patch level after you install an enhancement pack, but you will still be able to roll back any fixes that are installed on the new enhancement pack.

You might disable backups to speed up installation time, to save disk space, or because you have your own methods to back up the cluster.

Default

Y (backup and rollback are fully enabled)

LSF_ADMINS

Required. List of LSF administrators.

Syntax

```
LSF_ADMINS="user_name [ user_name ... ]"
```

Description

The first user account name in the list is the primary LSF administrator. It cannot be the root user account.

Typically this account is named `lsfadmin`. It owns the LSF configuration files and log files for job events. It also has permission to reconfigure LSF and to control batch jobs that are submitted by other users. It typically does not have authority to start LSF daemons. Usually, only root has permission to start LSF daemons.

All the LSF administrator accounts must exist on all hosts in the cluster before you install LSF. Secondary LSF administrators are optional.

Valid Values

Existing user accounts

Example

```
LSF_ADMINS="lsfadmin user1 user2"
```

Default

None - required variable

LSF_ENTITLEMENT_FILE

Full path to the LSF entitlement file.

Syntax

```
LSF_ENTITLEMENT_FILE=path
```

Description

LSF uses the entitlement to determine which feature set to enable or disable based on the edition of the product. The entitlement file for LSF Standard Edition is `lsf_std_entitlement.dat`. For LSF Express Edition, the file is `lsf_exp_entitlement.dat`. For LSF Advanced Edition, the file is `lsf_adv_entitlement.dat`. The entitlement file is installed as `<LSF_TOP>/conf/lsf.entitlement`.

You must download the entitlement file for the edition of the product you are running, and set **LSF_ENTITLEMENT_FILE** to the full path to the entitlement file you downloaded.

After LSF is installed and running, run the **lsid** command to see which edition of LSF is enabled.

Example

```
LSF_ENTITLEMENT_FILE=/usr/share/lsf_distrib/lsf.entitlement
```

Default

None - required variable

LSF_LIM_PORT

TCP service port for the server host.

Syntax

```
LSF_LIM_PORT="port_number"
```

Description

Use the same port number as the value of the **LSF_LIM_PORT** parameter in the `lsf.conf` file on the master host.

Default

7869

LSF_SERVER_HOSTS

Required for non-shared server host installation. This parameter defines a list of hosts that can provide host and load information to client hosts.

Syntax

```
LSF_SERVER_HOSTS="host_name [host_name ...]"
```

Description

If you do not define this parameter, clients contact the master LIM for host and load information. List of LSF server hosts in the cluster to be contacted.

Define this parameter for large clusters to decrease the load on the master LIM. Do not specify the master host in the list. Client commands query the LIMs on these defined hosts, which offload traffic from the master LIM.

Define this parameter to ensure that commands run successfully when no LIM is running on the local host, or when the local LIM is started.

Include the list of hosts that are defined in the **LSF_MASTER_LIST** parameter in the `lsf.conf` file; specify the primary master host last. For example,

```
LSF_MASTER_LIST="lsfmaster hostE"
```

```
LSF_SERVER_HOSTS="hostB hostC hostD hostE lsfmaster"
```

Specify a list of host names two ways:

- Host names that are separated by spaces
- Name of a file that contains a list of host names, one host per line.

Valid Values

Any valid LSF host name

Examples

The following definition specifies a list of host names:

```
LSF_SERVER_HOSTS="hosta hostb hostc hostd"
```

The following definition specifies a host list file:

```
LSF_SERVER_HOSTS=:lsf_server_hosts
```

The `lsf_server_hosts` file contains a list of hosts:

```
hosta hostb hostc hostd
```

Default

None

LSF_TARDIR

Full path to the directory that contains the LSF distribution `.tar` files.

Syntax

```
LSF_TARDIR="/path"
```

Example

```
LSF_TARDIR="/usr/local/lsf_distrib"
```

Default

The parent directory of the current working directory. For example, if the `lsfinstall` command is running under the `usr/share/lsf_distrib/lsf_lsfinstall` directory, the default value for the `LSF_TARDIR` parameter is `usr/share/lsf_distrib`.

LSF_LOCAL_RESOURCES

Defines instances of local resources on the server host.

Syntax

```
LSF_LOCAL_RESOURCES="resource ..."
```

Description

- For numeric resources, define name-value pairs:

```
"[resourcemap value*resource_name]"
```

- For Boolean resources, define the resource name in the form:

```
"[resource resource_name]"
```

When the server host calls the master host to add itself, it also reports its local resources. The local resources to be added must be defined in the `lsf.shared` file.

If the same resource is already defined in the `lsf.shared` file as default or all, it cannot be added as a local resource. The shared resource overrides the local one.

Tip:

The `LSF_LOCAL_RESOURCES` parameter is set in the `slave.config` file during installation. If the `LSF_LOCAL_RESOURCES` parameter is already defined in a local `lsf.conf` file on the server host, the `lsfinstall` command does not add resources that you define in the `LSF_LOCAL_RESOURCES` parameter in the `slave.config` file. Do not have duplicate `LSF_LOCAL_RESOURCES` entries in the `lsf.conf` file. If local resources are defined more than once, only the last definition is valid.

Important:

Resources must already be mapped to hosts in the ResourceMap section of the `lsf.cluster.cluster_name` file. If the ResourceMap section does not exist, local resources are not added.

Example

```
LSF_LOCAL_RESOURCES="[resourcemap 1*verilog] [resource linux]"
```

Default

None

LSF_TOP

Required. Full path to the top level LSF installation directory.

Syntax

```
LSF_TOP="/path"
```

Description

Important: You must use the same path for every server host you install.

Valid value

The **LSF_TOP** file path cannot be the root directory (/).

Example

```
LSF_TOP="/usr/local/lsf"
```

Default

None - required variable

SILENT_INSTALL

Enables silent installation.

Syntax

```
SILENT_INSTALL="Y" | "N"
```

Description

Enabling the silent installation (that is, setting this parameter to Y) means that you want to do the silent installation and accept the license agreement.

Default

N

LSF_SILENT_INSTALL_TARLIST

A string that contains all LSF package names to be installed.

Syntax

```
LSF_SILENT_INSTALL_TARLIST="ALL" | "Package_Name ..."
```

Description**Example**

This name list applies only to the silent installation mode. Supports keywords all, ALL, and All, which can install all packages in the **LSF_TARDIR** directory.

```
LSF_SILENT_INSTALL_TARLIST="ALL" | "lsf10.1.0.9_linux2.6-glibc2.3-x86_64.tar.Z"
```

Default

None

Chapter 2. Environment variables

Learn how LSF environment variables are set for job execution, job resize notification command, and for session scheduler (**ssched**).

Environment variables set for job execution

In addition to environment variables inherited from the user environment, LSF also sets several other environment variables for batch jobs.

LSF transfers most environment variables between submission and execution hosts.

Environment variables related to file names and job spooling directories support paths that contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

Environment variables related to command names and job names can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

- **LSB_ERRORFILE**: Name of the error file specified with a **bsub -e**.
- **LSB_EFFECTIVE_RSRCREQ**: The job effective resource requirement. If the job level resource requirement was changed by the **bmod -R** command for a running job, the changed effective resource requirement is not available through this environment variable.
- **LSB_JOBID**: Job ID assigned by LSF.
- **LSB_JOBINDEX**: Index of the job that belongs to a job array.
- **LSB_CHKPNT_DIR**: This variable is set each time a checkpointed job is submitted. The value of the variable is *chkpnt_dir/job_Id*, a subdirectory of the checkpoint directory that is specified when the job is submitted. The subdirectory is identified by the job ID of the submitted job.
- **LSB_HOSTS**: The list of hosts that are used to run the batch job. For sequential jobs, this is only one host name. For parallel jobs, this includes multiple host names.
- **LSB_RESIZABLE**: Indicates that a job is resizable or auto-resizable.
- **LSB_QUEUE**: The name of the queue the job is dispatched from.
- **LSB_JOBNAME**: Name of the job.
- **LSB_RESTART**: Set to 'Y' if the job is a restarted job or if the job has been migrated. Otherwise this variable is not defined.
- **LSB_EXIT_PRE_ABORT**: Set to an integer value representing an exit status. A pre-execution command should exit with this value if it wants the job to be aborted instead of requeued or executed.
- **LSB_EXIT_REQUEUE**: Set to the **REQUEUE_EXIT_VALUES** parameter of the queue. This variable is not defined if **REQUEUE_EXIT_VALUES** is not configured for the queue.
- **LSB_INTERACTIVE**: Set to 'Y' if the job is submitted with the **-I** option. Otherwise, it is not defined.
- **LS_EXECCWD**: Sets the current working directory for job execution.
- **LS_JOBPID**: Set to the process ID of the job.
- **LS_SUBCWD**: This is the directory on the submission when the job was submitted. This is different from **PWD** only if the directory is not shared across machines or when the execution account is different from the submission account as a result of account mapping.
- **LSB_BIND_JOB**: Set to the value of binding option. But when the binding option is **USER**, **LSB_BIND_JOB** is set to the real binding decision of end user.

Note:

If the binding option is **Y**, **LSB_BIND_JOB** is set to **BALANCE**. If the binding option is **N**, **LSB_BIND_JOB** is set to **NONE**.

Environment variables for job resize notification

- **LSB_BIND_CPU_LIST**: Set to the actual CPU list used when the job is sequential job and single host parallel job.

If the job is a multi-host parallel job, **LSB_BIND_CPU_LIST** is set to the value in submission environment variable `$LSB_USER_BIND_CPU_LIST`. If there is no such submission environment variable in user's environment, **LSB_BIND_CPU_LIST** is set to an empty string.

The following environment variables are set only in the post job environment:

- **LSB_ACCUMULATED_CPU_TIME**: Job accumulated CPU time. For migrated jobs, the CPU time can be accumulated across migration runs. Job CPU time is shown to two decimal places.
- **LSB_MAX_MEM_RUSAGE**: Maximum memory rusage of the job processes, not including `post_exec`. Always in KB.
- **LSB_MAX_SWAP_RUSAGE**: Maximum swap rusage of the job processes, not including `post_exec`. Always in KB.
- **LSB_MAX_PROCESSES_RUSAGE**: Number of processes for the job, not including `post_exec`.
- **LSB_MAX_THREADS_RUSAGE**: Number of threads for the job, not including `post_exec`.
- **LSB_JOB_SUBMIT_TIME**: The time that the job was submitted.
- **LSB_JOB_START_TIME**: The time that the job was started. For requeued or migrated jobs, the start time is the time the job started after it was requeued or migrated. For chunk job members, it is the time the member actually starts, not the start time of the chunk.
- **LSB_JOB_END_TIME**: The time that the job ended, not including `post_exec`.
- **LSB_JOB_PEND_TIME**: Pend time for the job, in seconds, which is calculated from submit time and start time (start time - submit time). For a requeued or migrated job the pend time may be longer than its real time in `PEND` state, including the time for the previous run. In those cases, the pend time is the time from job submission to the time of last job start..
- **LSB_JOB_STATUS**: Job status value as defined in `lsbatch.h`. **LSB_JOB_STATUS** is set to 32 for job exit and 64 for job doneb.

Environment variables for resize notification command

All environment variables that are set for a job are also set when a job is resized.

The following (additional) environment variables apply only to the resize notification command environment (when using resizable jobs):

- **LSB_RESIZE_NOTIFY_OK**: A notification command should exit with this variable if the allocation resize notification command succeeds.

LSF updates the job allocation to reflect the new allocation.

- **LSB_RESIZE_NOTIFY_FAIL**: A notification command should exit with this variable if the allocation resize notification command fails.

For an allocation grow event, LSF schedules the pending allocation request.

For an allocation shrink event, LSF fails the release request.

- **LSB_RESIZE_EVENT = grow | shrink**: Indicates why the notification command was called. Grow means add more resources to an existing allocation. Shrink means remove some resources from existing allocation.
- **LSB_RESIZE_HOSTS = hostA numA hostB numB ... hostZ numZ**: Lists the additional slots for a grow event, or the released slots for a shrink event.

Environment variables for session scheduler (ssched)

By default, all environment variables that are set as part of the session are available in each task's execution environment.

Variables for the execution host of each task

The following environment variables are reset according to the execution host of each task:

- EGO_SERVERDIR
- LSB_TRAPSIGS
- LSF_SERVERDIR
- HOSTTYPE
- LSB_HOSTS
- LSF_BINDIR
- EGO_BINDIR
- PWD
- HOME
- LSB_ERRORFILE
- LSB_OUTPUTFILE
- TMPDIR
- LSF_LIBDIR
- EGO_LIBDIR
- LSB_MCPU_HOSTS
- PATH (prepend LSF_BINDIR)
- LD_LIBRARY_PATH (prepend LSF_LIBDIR and EGO_LIBDIR)

Environment variables NOT available in the task environment

- LSB_JOBRES_PID
- LSB_EEXEC_REAL_UID
- LS_EXEC_T
- LSB_INTERACTIVE
- LSB_CHKFILENAME
- SPOOLDIR
- LSB_ACCT_FILE
- LSB_EEXEC_REAL_GID
- LSB_CHKPNT_DIR
- LSB_CHKPNT_PERIOD
- LSB_JOB_STARTER
- LSB_EXIT_REQUEUE
- LSB_DJOB_RU_INTERVAL
- LSB_DJOB_HB_INTERVAL
- LSB_DJOB_HOSTFILE
- LSB_JOBEXIT_INFO
- LSB_JOBPEND
- LSB_EXECHOSTS

Environment variables corresponding to the session job

- LSB_JOBID
- LSB_JOBINDEX

Environment variable reference

- LSB_JOBINDEX_STEP
- LSB_JOBINDEX_END
- LSB_JOBPID
- LSB_JOBNAME
- LSB_JOBFILENAME

Environment variables set individually for each task

- LSB_TASKID—The current task ID
- LSB_TASKINDEX—The current task index

Environment variables for data provenance

All environment variables that are set for a job are also set when data provenance is enabled for a job.

The following additional environment variables apply only to the data provenance environment (that is, the following environment variables are available to the predefined `tag.sh` script that is used for data provenance):

- **LSB_DP_SUBCMD**: The **bsub** job submission command.
- **LSB_DP_STDINFILE**: The standard input file for the job, as defined in the **bsub -i** option.
- **LSB_DP_SUBFILES_index**: The source files on the submission host (to be copied to the execution host), as defined in the **bsub -f** option.
- **LSB_DP_EXECFILES_index**: The destination files on the execution host (copied from the submission host), as defined in the **bsub -f** option.
- **LSB_DP_FILES**: The number of files to be copied, as defined in the **bsub -f** option.
- **LSB_DP_INPUTFILES_index**: The files that are defined in the **esub.dprov** script.
- **LSB_DP_INPUTFILES**: The number files that are defined in the **esub.dprov** script.

Environment variable reference

Reference for LSF environment variables.

BSUB_BLOCK

The `BSUB_BLOCK` variable is set when you submit a job for synchronous job execution with the `bsub -K` option.

Description

If set, tells NIOS that the job is running in batch mode.

Default

Not defined

Notes

Synchronous execution means you have to wait for the job to finish before you can continue.

Where defined

Set internally

See also

The -K option of **bsub**

BSUB_CHK_RESREQ

When the `BSUB_CHK_RESREQ` variable is set, **bsub** checks the syntax of the resource requirement selection string without actually submitting the job for scheduling and dispatch.

Syntax

`BSUB_CHK_RESREQ=any_value`

Description

Use `BSUB_CHK_RESREQ` to check the compatibility of your existing resource requirement select strings against the stricter syntax enabled by the `LSF_STRICT_RESREQ=y` parameter in the `lsf.conf` file. The `LSF_STRICT_RESREQ` parameter does not need to be set to check the resource requirement selection string syntax.

The **bsub** command only checks the select section of the resource requirement. Other sections in the resource requirement string are not checked.

Default

Not defined

Where defined

From the command line

Example

```
BSUB_CHK_RESREQ=1
```

BSUB_QUIET**Syntax**

`BSUB_QUIET=any_value`

Description

Controls the printing of information about job submissions. If set, **bsub** will not print any information about job submission. For example, it will not print `<Job is submitted to default queue <normal>, nor <Waiting for dispatch>`.

Default

Not defined

Where defined

From the command line

Example

```
BSUB_QUIET=1
```

BSUB_QUIET2

Syntax

BSUB_QUIET2=*any_value*

Description

Suppresses the printing of information about job completion when a job is submitted with the **bsub** -K option.

If set, **bsub** will not print information about job completion to stdout. For example, when this variable is set, the message <<Job is finished>> will not be written to stdout.

If BSUB_QUIET and BSUB_QUIET2 are both set, no job messages will be printed to stdout.

Default

Not defined

Where defined

From the command line

Example

```
BSUB_QUIET2=1
```

BSUB_STDERR

Syntax

BSUB_STDERR=y

Description

Redirects LSF messages for **bsub** to stderr.

By default, when this parameter is not set, LSF messages for **bsub** are printed to stdout.

When this parameter is set, LSF messages for **bsub** are redirected to stderr.

Default

Not defined

Where defined

From the command line on UNIX. For example, in **csH**:

```
setenv BSUB_STDERR Y
```

From the Control Panel on Windows, as an environment variable

CLEARCASE_DRIVE

Syntax

CLEARCASE_DRIVE=*drive_letter*:

Description

Optional, Windows only.

Defines the virtual drive letter for a Rational ClearCase view to the drive. This is useful if you wish to map a Rational ClearCase view to a virtual drive as an alias.

If this letter is unavailable, Windows attempts to map to another drive. Therefore, CLEARCASE_DRIVE only defines the default drive letter to which the Rational ClearCase view is mapped, not the final selected drive letter. However, the PATH value is automatically updated to the final drive letter if it is different from CLEARCASE_DRIVE.

Notes:

CLEARCASE_DRIVE is not case sensitive.

Where defined

From the command line

Example

```
CLEARCASE_DRIVE=F:
```

```
CLEARCASE_DRIVE=f:
```

See also

CLEARCASE_MOUNTDIR, CLEARCASE_ROOT

CLEARCASE_MOUNTDIR**Syntax**

```
CLEARCASE_MOUNTDIR=path
```

Description

Optional.

Defines the Rational ClearCase mounting directory.

Default

```
/vobs
```

Notes:

CLEARCASE_MOUNTDIR is used if any of the following conditions apply:

- A job is submitted from a UNIX environment but run in a Windows host.
- The Rational ClearCase mounting directory is not the default /vobs

Where defined

From the command line

Example

```
CLEARCASE_MOUNTDIR=/myvobs
```

See also

CLEARCASE_DRIVE, CLEARCASE_ROOT

CLEARCASE_ROOT

Syntax

```
CLEARCASE_ROOT=path
```

Description

The path to the Rational ClearCase view.

In Windows, this path must define an absolute path starting with the default ClearCase drive and ending with the view name without an ending backslash (\).

Notes

CLEARCASE_ROOT must be defined if you want to submit a batch job from a ClearCase view.

For interactive jobs, use `bsub -I` to submit the job.

Where defined

In the job starter, or from the command line

Example

In UNIX:

```
CLEARCASE_ROOT=/view/myview
```

In Windows:

```
CLEARCASE_ROOT=F:\myview
```

See also

CLEARCASE_DRIVE, CLEARCASE_MOUNTDIR, LSF_JOB_STARTER

DAEMON_WRAP_DEBUG

LSF integration for Rational ClearCase. Enables daemon wrapper debugging.

Syntax

```
DAEMON_WRAP_DEBUG=1
```

Description

When DAEMON_WRAP_DEBUG is set to 1, LSF logs the ClearCase stdout and stderr streams to `/tmp/daemons.wrap.log`. Use one of the following shell commands to set the DAEMON_WRAP_DEBUG environment variable:

- In **csh** or **tcsh**:

```
setenv DAEMON_WRAP_DEBUG 1
```

- In **sh**, **ksh**, or **bash**:

```
export DAEMON_WRAP_DEBUG=1
```

ELIM_ABORT_VALUE

Syntax

ELIM_ABORT_VALUE

Description

Used when writing an **elim** executable to test whether the **elim** should run on a particular host. If the host does not have or share any of the resources listed in the environment variable **LSF_RESOURCES**, your **elim** should exit with **\$ELIM_ABORT_VALUE**.

When the MELIM finds an **elim** that exited with **ELIM_ABORT_VALUE**, the MELIM marks the **elim** and does not restart it on that host.

Where defined

Set by the master **elim** (MELIM) on the host when the MELIM invokes the **elim** executable

LS_EXEC_T

Syntax

LS_EXEC_T=START | END | CHKPNT | JOB_CONTROLS

Description

Indicates execution type for a job. LS_EXEC_T is set to:

- START or END for a job when the job begins executing or when it completes execution
- CHKPNT when the job is checkpointed
- JOB_CONTROLS when a control action is initiated

Where defined

Set by sbatchd during job execution

LS_JOBPID

Description

The process ID of the job.

Where defined

During job execution, sbatchd sets LS_JOBPID to be the same as the process ID assigned by the operating system.

LS_SUBCWD

Description

The current working directory (cwd) of the submission host where the remote task command was executed.

Environment variable reference

The current working directory can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows.

How set

1. LSF looks for the PWD environment variable. If it finds it, sets LS_SUBCWD to PWD.
2. If the PWD environment variable does not exist, LSF looks for the CWD environment variable. If it finds CWD, sets LS_SUBCWD to CWD.
3. If the CWD environment variable does not exist, LSF calls the `getwd()` system function to retrieve the current working directory path name. LSF sets LS_SUBCWD to the value that is returned.

Where defined

Set by sbatchd

LSB_AFFINITY_HOSTFILE

Syntax

LSB_AFFINITY_HOSTFILE=*file_path*

Description

Path to the NUMA CPU and memory affinity binding decision file.

On the first execution host, LSF **sbatchd** will create a binding decision file per job under the same location as \$LSB_DJOB_HOSTFILE. The binding decision file has a format similar to the job Host File, one task per line.

Each line includes: *host_name cpu_id_list NUMA_node_id_list memory_policy*.

For memory policy, 1 means `localonly`, 2 means `localprefer`, as specified in the affinity resource requirement `membind` parameter.

Comma (,) is the only supported delimiter for the list of CPU IDs and the list of NUMA node IDs.

The following Host File represents a job with 6 tasks:

- Host1 and Host2 each have two tasks bound to CPUs {0,1,2,3} and {4,5,6,7}, and NUMA nodes 0 and 1 respectively with a `membind=localprefer` policy.
- Host3 and Host4 each have one task bound to CPUs {0,1,2,3} and NUMA node 0, again with a `membind=localprefer` policy.

```
Host1 0,1,2,3 0 2
Host1 4,5,6,7 1 2
Host2 0,1,2,3 0 2
Host2 4,5,6,7 1 2
Host3 0,1,2,3 0 2
Host4 0,1,2,3 0 2
```

Default

Not defined

Where defined

Set during job execution.

LSB_BHIST_HOURS

Usage

LSB_BHIST_HOURS=*hours*

Description

The multiline format of **bhist** considers job events from the last LSB_BHIST_HOURS.

Use **LSB_BHIST_HOURS** with the command **bhist -t** to display job events before the current time for a longer or shorter period than the default one week.

Default

Not defined

If not defined, the multiline format for **bhist** considers job events from the last week.

LSB_BIND_CPU_LIST

Syntax

LSB_BIND_CPU_LIST=*cpu_list*

Description

LSB_BIND_CPU_LIST contains allocated CPUs on each host. LSF will combine all of the allocated CPUs for all the tasks that ended up on the host (for the given job), and set this environment variable to the entire list before launching tasks. The following example corresponds to the tasks specified in the example for LSB_AFFINITY_HOSTFILE:

```

LSB_BIND_CPU_LIST on Host1: 0,1,2,3,4,5,6,7
LSB_BIND_CPU_LIST on Host2: 0,1,2,3,4,5,6,7
LSB_BIND_CPU_LIST on Host3: 0,1,2,3
LSB_BIND_CPU_LIST on Host4: 0,1,2,3

```

Default

Not defined

Where defined

Set during job execution.

LSB_BIND_MEM_LIST

Syntax

LSB_BIND_MEM_LIST=*memory_node_list*

Description

LSB_BIND_MEM_LIST contains allocated memory nodes on each host. If the job is submitted with a memory affinity requirement, LSF will combine all of the allocated NUMA node IDs for all the tasks that ended up on the host (for the given job), and set this environment variable to the entire list before launching tasks. The following example corresponds to the tasks specified in the example for LSB_AFFINITY_HOSTFILE:

```

LSB_BIND_MEM_LIST on Host1: 0,1
LSB_BIND_MEM_LIST on Host2: 0,1

```

Environment variable reference

```
LSB_BIND_MEM_LIST on Host3: 0  
LSB_BIND_MEM_LIST on Host4: 0
```

Default

Not defined

Where defined

Set during job execution.

LSB_BIND_MEM_POLICY

Syntax

LSB_BIND_MEM_POLICY=localprefer | localonly

Description

For jobs submitted with a NUMA memory affinity resource requirement, LSF sets the memory binding policy in LSB_BIND_MEM_POLICY environment variable, as specified in the membind affinity resource requirement parameter: either `localprefer` or `localonly`.

Default

Not defined

Where defined

Set during job execution.

LSB_BHOSTS_FORMAT

This parameter can be set from the command line or from `lsf.conf`.

See LSB_BHOSTS_FORMAT in `lsf.conf`.

LSB_BJOBS_FORMAT

This parameter can be set from the command line or from `lsf.conf`.

See LSB_BJOBS_FORMAT in `lsf.conf`.

LSB_BMGROUP_ALLREMOTE_EXPAND

Syntax

This parameter can be set from the command line or from `lsf.conf`.

See LSB_BMGROUP_ALLREMOTE_EXPAND in `lsf.conf`.

LSB_BQUEUES_FORMAT

This parameter can be set from the command line or from `lsf.conf`.

See LSB_BQUEUES_FORMAT in `lsf.conf`.

LSB_BSUB_ERR_RETRY

Syntax

LSB_BSUB_ERR_RETRY=RETRY_CNT[*number*] ERR_TYPE[error1 error2 ...]

Description

In some cases, jobs can benefit from being automatically retried in the case of failing for a particular error. When specified, **LSB_BSUB_ERR_RETRY** automatically retries jobs that exit with a particular reason, up to the number of times specified by **RETRY_CNT**.

Only the following error types (**ERR_TYPE**) are supported:

- **BAD_XDR**: Error during XDR.
- **MSG_SYS**: Failed to send or receive a message.
- **INTERNAL**: Internal library error.

The number of retries (**RETRY_CNT**) can be a minimum of 1 to a maximum of 50.

Considerations when setting this parameter:

- Users may experience what seems like a lag during job submission while the job is retried automatically in the background.
- Users may see a job submitted more than once, with no explanation (no error is communicated to the user; the job keeps getting submitted until it succeeds or reaches its maximum retry count). In this case, the job ID also changes each time the error is retried.

Default

N

LSB_BUSERS_FORMAT

This parameter can be set from the command line or from `lsf.conf`.

See **LSB_BUSERS_FORMAT** in `lsf.conf`.

LSB_CHKPNT_DIR**Syntax**

`LSB_CHKPNT_DIR=checkpoint_dir/job_ID`

Description

The directory containing files related to the submitted checkpointable job.

Valid values

The value of `checkpoint_dir` is the directory you specified through the `-k` option of **bsub** when submitting the checkpointable job.

The value of `job_ID` is the job ID of the checkpointable job.

Where defined

Set by LSF, based on the directory you specified when submitting a checkpointable job with the `-k` option of **bsub**.

LSB_CONTAINER_IMAGE_UPDATE**Syntax**

`LSB_CONTAINER_IMAGE_UPDATE=Y | y`

Environment variable reference

Description

If defined, LSF updates the Docker image that is specified in the job related application file before execution.

Default

If undefined, the Docker image does not update.

LSB_DATA_CACHE_TOP

Contains a string defining the location of the data management staging area relative to the compute node. The value of this variable is equivalent to **STAGING_AREA** in `lsf.datamanager`.

See also

`lsf.datamanager`

LSB_DATA_META_FILE

For jobs submitted with data requirements, this variable contains a string defining the location of the job's metadata file relative to the compute node. The value of this variable is equivalent to the value of `STAGING_AREA/work/cluster_name/jobID/stgin.meta`.

See also

LSB_OUTDIR, the `bsub -data` option

LSB_DATA_PROVENANCE

Syntax

`LSB_DATA_PROVENANCE=Y | y`

Description

Enables data provenance for LSF jobs, which allows you to trace files that are generated by LSF jobs. Data provenance information also allows you to reproduce your data results when using the same job input and steps.

You must use IBM Spectrum Scale as the file system to support the extended attribute specification of files that is required for data provenance. These attributes are attached to the files, which allows LSF to maintain the files with the provenance data.

Run the predefined post execution script `tag.sh` with this environment variable in the job submission command to use data provenance for a particular job. For example, `bsub -e 'LSB_DATA_PROVENANCE=y' -Ep 'tag.sh' myjob`

This environment variable is automatically enabled in the **esub.dprov** application.

Default

N

Where defined

Set by **esub.dprov** or **bsub** at job submission.

LSB_DEBUG

This parameter can be set from the command line or from `lsf.conf`. See `LSB_DEBUG` in `lsf.conf`.

LSB_DEBUG_CMD

This parameter can be set from the command line or from `lsf.conf`. See `LSB_DEBUG_CMD` in `lsf.conf`.

LSB_DEBUG_MBD

This parameter can be set from the command line with `badadmin mbddebug` or from `lsf.conf`. See `LSB_DEBUG_MBD` in `lsf.conf`.

LSB_DEBUG_SBD

This parameter can be set from the command line with `badadmin sbddebug` or from `lsf.conf`. See `LSB_DEBUG_SBD` in `lsf.conf`.

LSB_DEBUG_SCH

This parameter can be set from the command line or from `lsf.conf`. See `LSB_DEBUG_SCH` in `lsf.conf`.

LSB_DEFAULT_JOBGROUP

Syntax

`LSB_DEFAULT_JOBGROUP=job_group_name`

Description

The name of the default job group.

When you submit a job to LSF without explicitly specifying a job group, LSF associates the job with the specified job group. `LSB_DEFAULT_JOBGROUP` overrides the setting of `DEFAULT_JOBGROUP` in `lsb.params`. The `bsub -g job_group_name` option overrides both `LSB_DEFAULT_JOBGROUP` and `DEFAULT_JOBGROUP`.

If you submit a job without the `-g` option of `bsub`, but you defined `LSB_DEFAULT_JOBGROUP`, then the job belongs to the job group specified in `LSB_DEFAULT_JOBGROUP`.

Job group names must follow this format:

- Job group names must start with a slash character (/). For example, `LSB_DEFAULT_JOBGROUP=/A/B/C` is correct, but `LSB_DEFAULT_JOBGROUP=A/B/C` is not correct.
- Job group names cannot end with a slash character (/). For example, `LSB_DEFAULT_JOBGROUP=/A/` is not correct.
- Job group names cannot contain more than one slash character (/) in a row. For example, job group names like `LSB_DEFAULT_JOBGROUP=/A//B` or `LSB_DEFAULT_JOBGROUP=A///B` are not correct.
- Job group names cannot contain spaces. For example, `LSB_DEFAULT_JOBGROUP=/A/B C/D` is not correct.
- Project names and user names used for macro substitution with `%p` and `%u` cannot start or end with slash character (/).
- Project names and user names used for macro substitution with `%p` and `%u` cannot contain spaces or more than one slash character (/) in a row.
- Project names or user names containing slash character (/) will create separate job groups. For example, if the project name is `canada/projects`, `LSB_DEFAULT_JOBGROUP=%p` results in a job group hierarchy `/canada/projects`.

Environment variable reference

Where defined

From the command line

Example

```
LSB_DEFAULT_JOBGROUP=/canada/projects
```

Default

Not defined

See also

DEFAULT_JOBGROUP in `lsb.params`, the **-g** option of **bsub**

LSB_DEFAULTPROJECT

Syntax

```
LSB_DEFAULTPROJECT=project_name
```

Description

The name of the project to which resources consumed by a job will be charged.

Default

Not defined

Notes

Project names can be up to 511 characters long.

If the LSF administrator defines a default project in the `lsb.params` configuration file, the system uses this as the default project. You can change the default project by setting **LSB_DEFAULTPROJECT** or by specifying a project name with the **-P** option of **bsub**.

If you submit a job without the **-P** option of **bsub**, but you defined **LSB_DEFAULTPROJECT**, then the job belongs to the project specified in **LSB_DEFAULTPROJECT**.

If you submit a job with the **-P** option of **bsub**, the job belongs to the project specified through the **-P** option.

Where defined

From the command line, or through the **-P** option of **bsub**

Example

```
LSB_DEFAULTPROJECT=engineering
```

See also

DEFAULT_PROJECT in `lsb.params`, the **-P** option of **bsub**

LSB_DEFAULTQUEUE

Syntax

LSB_DEFAULTQUEUE=*queue_name*

Description

Defines the default LSF queue.

Default

mbatchd decides which is the default queue. You can override the default by defining **LSB_DEFAULTQUEUE**.

Notes

If the LSF administrator defines a default queue in the `lsb.params` configuration file, then the system uses this as the default queue. Provided you have permission, you can change the default queue by setting **LSB_DEFAULTQUEUE** to a valid queue (see **bqueues** for a list of valid queues).

Where defined

From the command line

See also

DEFAULT_QUEUE in `lsb.params`

LSB_DJOB_COMMFAIL_ACTION

Syntax

LSB_DJOB_COMMFAIL_ACTION="KILL_TASKS"

Description

Defines the action LSF should take if it detects a communication failure with one or more remote parallel or distributed tasks. If defined, LSF will try to kill all the current tasks of a parallel or distributed job associated with the communication failure. If not defined, the job RES notifies the task RES to terminate all tasks, and shut down the entire job.

Default

Terminate all tasks, and shut down the entire job

Valid values

KILL_TASKS

Where defined

Set by the system based on the value of the parameter `DJOB_COMMFAIL_ACTION` in `lsb.applications` when running **bsub -app** for the specified application

See also

`DJOB_COMMFAIL_ACTION` in `lsb.applications`

LSB_DJOB_ENV_SCRIPT

Syntax

LSB_DJOB_ENV_SCRIPT=*script_name*

Description

Defines the name of a user-defined script for setting and cleaning up the parallel or distributed job environment. This script will be executed by LSF with the argument `setup` before launching a parallel or distributed job, and with argument `cleanup` after the parallel job is finished.

The script will run as the user, and will be part of the job.

If a full path is specified, LSF will use the path name for the execution. Otherwise, LSF will look for the executable from \$LSF_BINDIR.

Where defined

Set by the system to the value of the parameter `DJOB_ENV_SCRIPT` in `lsb.applications` when running **bsub -app** for the specified application

See also

`DJOB_ENV_SCRIPT` in `lsb.applications`

LSB_DJOB_HB_INTERVAL

Syntax

LSB_DJOB_HB_INTERVAL=*seconds*

Description

Defines the time interval between heartbeat messages sent by the remote execution tasks to the head node. If the head node does not receive a heartbeat message from one task within 2 intervals, LSF will take action according to how **LSB_DJOB_COMMFAIL_ACTION** is specified. Heartbeat message sending cannot be disabled. **LSB_DJOB_HB_INTERVAL** can be set as an environment variable of **bsub**. If defined, it will overwrite **DJOB_HB_INTERVAL** configuration in the application profile. If neither parameter is defined, LSF uses the default value to report heartbeat messages. The default value is calculated with the following formula:

$$\text{MAX}(60, \text{number_of_execution_hosts} * 0.12)$$

Valid values must be positive integers.

LSB_DJOB_RU_INTERVAL

Syntax

LSB_DJOB_RU_INTERVAL=*seconds*

Description

Defines the time interval that LSF reports parallel job rusage on each execution host. A value 0 seconds means disable the resource update. **LSB_DJOB_RU_INTERVAL** can be set as an environment variable of **bsub**. If defined, it overwrites the **DJOB_RU_INTERVAL** configuration in application profile. If neither **LSB_DJOB_RU_INTERVAL** nor **DJOB_RU_INTERVAL** is defined, LSF will use the default value to report resource usage. The default value is calculate with the following formula:

$$\text{MAX}(60, \text{number_of_execution_hosts} * 0.3)$$

Valid values are non-negative integers.

LSB_DJOB_PE_NETWORK

Description

Network resource information for IBM Parallel Environment (PE) jobs submitted with the `bsub -network` option, or to a queue (defined in `lsb.queues`) or an application profile (defined in `lsb.applications`) with the `NETWORK_REQ` parameter defined.

Where defined

Set by `sbatchd` before a job is dispatched.

LSB_DJOB_NUMPROC

Syntax

`LSB_DJOB_NUMPROC=num`

Description

The number of processors (slots) allocated to the job.

Default

Not defined

Where defined

Set by `sbatchd` before starting a job on the execution host.

See Also

LSB_MCPU_HOSTS

LSB_DJOB_RANKFILE

Syntax

`LSB_DJOB_RANKFILE=file_path`

Description

When a job is submitted (`bsub -hostfile`) or modified (`bmod -hostfile`) with a user-specified host file, the `LSB_DJOB_RANKFILE` environment variable is generated from the user-specified host file. If a job is not submitted with a user-specified host file then `LSB_DJOB_RANKFILE` points to the same file as `LSB_DJOB_HOSTFILE`.

Duplicate host names are combined, along with the total number of slots for a host name and the results are used for scheduling (`LSB_DJOB_HOSTFILE` groups the hosts together) and for `LSB_MCPU_HOSTS`. `LSB_MCPU_HOSTS` represents the job allocation.

The **esub** parameter `LSB_SUB4_HOST_FILE` reads and modifies the value of the **-hostfile** option.

A host name repeated sequentially will be combined to one host name with a summary of slots but host name order will be maintained.

For example, if the user specified host file contains:

```
host01
host01 2
```

```
host02
host01
host02
host02
host03
```

then the **bjobs** and **bhist** commands will show the following allocation summary for the job:

```
USER-SPECIFIED HOST FILE:
HOST           SLOTS
host01         3
host02         1
host01         1
host02         2
host03         1
```

Default

By default, points to LSB_DJOB_HOSTFILE.

LSB_DJOB_TASK_BIND

Syntax

```
LSB_DJOB_TASK_BIND=Y | y | N | n
```

Description

For CPU and memory affinity scheduling jobs launched with the **blaunch** distributed application framework.

If LSB_DJOB_TASK_BIND=Y in the submission environment before submitting the job, you must use **blaunch** to start tasks to enable LSF to bind each task to the proper CPUs or NUMA nodes. Only the CPU and memory bindings allocated to the task itself will be set in each task's environment.

If LSB_DJOB_TASK_BIND=N, or it is not set, each task will have the same CPU or NUMA node binding on one host.

If you do not use **blaunch** to start tasks, and use another MPI mechanism such as IBM Platform MPI or IBM Parallel Environment, you should not set DJOB_TASK_BIND or set it to N.

Default

N

Description

Where defined

Set by **sbatchd** before a job is dispatched.

LSB_DOCKER_IMAGE_AFFINITY

Syntax

```
LSB_DOCKER_IMAGE_AFFINITY=Y | y | N | n
```

Description

When scheduling Docker-based containerized jobs, setting this parameter to y or Y enables LSF to give preference for execution hosts that already have the requested Docker image. This reduces network bandwidth and the job start time because the execution host does not have to pull the Docker image from the repository and the job can immediately start on the execution host.

When this feature is enabled, LSF considers Docker image location information when scheduling Docker jobs. Docker image affinity interacts with host preference and **order[]** string requests in the following manner:

- If host preference is specified, the host preference is honored first. Among hosts with the same preference level, hosts with the requested Docker image are given higher priority.
- If the **order[]** string is specified, the hosts with the requested Docker image have a higher priority first. Among hosts that all have the requested Docker image, the **order[]** string is then honored.

Default

Not defined.

LSB_ECHKPNT_METHOD

This parameter can be set as an environment variable and/or in `lsf.conf`. See `LSB_ECHKPNT_METHOD` in `lsf.conf`.

LSB_ECHKPNT_METHOD_DIR

This parameter can be set as an environment variable and/or in `lsf.conf`. See `LSB_ECHKPNT_METHOD_DIR` in `lsf.conf`.

LSB_ECHKPNT_KEEP_OUTPUT

This parameter can be set as an environment variable and/or in `lsf.conf`. See `LSB_ECHKPNT_KEEP_OUTPUT` in `lsf.conf`.

LSB_EFFECTIVE_RSRCREQ

Syntax

`LSB_EFFECTIVE_RSRCREQ=res_req_string`

Description

The job effective resource requirement. If the job level resource requirement was changed by **bmod -R** for a running job, the changed effective resource requirement is not available via this environment variable.

Where defined

Set by **bsub** at job submission.

LSB_ERESTART_USRCMD

Syntax

`LSB_ERESTART_USRCMD=command`

Description

Original command used to start the job.

This environment variable is set by **erestart** to pass the job's original start command to a custom **erestart** method **erestart.method_name**. The value of this variable is extracted from the job file of the checkpointed job.

Environment variable reference

If a job starter is defined for the queue to which the job was submitted, the job starter is also included in `LSB_ERESTART_USRCMD`. For example, if the job starter is `/bin/sh -c "%USRCMD"` in `lsb.queues`, and the job name is **myapp -d**, `LSB_ERESTART_USRCMD` will be set to:

```
/bin/sh -c "myapp -d"
```

Where defined

Set by **erestart** as an environment variable before a job is restarted

See also

`LSB_ECHKPNT_METHOD`, **erestart**, **echkpnt**

LSB_EXEC_RUSAGE

Syntax

```
LSB_EXEC_RUSAGE="resource_name1 resource_value1 resource_name2 resource_value2..."
```

Description

Indicates which `rusage` string is satisfied to permit the job to run. This environment variable is necessary because the OR (|) operator specifies alternative `rusage` strings for running jobs.

Valid values

`resource_value1`, `resource_value2`,... refer to the resource values on `resource_name1`, `resource_name2`,... respectively.

Default

Not defined

Where defined

Set by LSF after reserving a resource for the job.

LSB_EXECHOSTS

Description

A list of hosts on which a batch job will run.

Where defined

Set by **sbatchd**

Product

MultiCluster

LSB_EXIT_IF_CWD_NOTEXIST

Syntax

```
LSB_EXIT_IF_CWD_NOTEXIST=Y|y|N|n
```

Description

Indicates that the job will exit if the current working directory specified by **bsub -cwd** or **bmod -cwd** is not accessible on the execution host.

Default

Not defined

Where defined

From the command line

LSB_EXIT_PRE_ABORT**Description**

The queue-level or job-level `pre_exec_command` can exit with this value if the job is to be aborted instead of being requeued or executed

Where defined

Set by `sbatchd`

See also

See `PRE_EXEC` in `lsb.queues`, or the `-E` option of **bsub**

LSB_EXIT_REQUEUE**Syntax**

```
LSB_EXIT_REQUEUE="exit_value1 exit_value2..."
```

Description

Contains a list of exit values found in the queue's `REQUEUE_EXIT_VALUES` parameter defined in `lsb.queues`.

Valid values

Any positive integers

Default

Not defined

Notes

If `LSB_EXIT_REQUEUE` is defined, a job will be requeued if it exits with one of the specified values.

`LSB_EXIT_REQUEUE` is not defined if the parameter `REQUEUE_EXIT_VALUES` is not defined.

Where defined

Set by the system based on the value of the parameter `REQUEUE_EXIT_VALUES` in `lsb.queues`

Example

```
LSB_EXIT_REQUEUE="7 31"
```

See also

REQUEUE_EXIT_VALUES in `lsb.queues`

LSB_FRAMES

Syntax

`LSB_FRAMES=start_number,end_number,step`

Description

Determines the number of frames to be processed by a frame job.

Valid values

The values of *start_number*, *end_number*, and *step* are positive integers. Use commas to separate the values.

Default

Not defined

Notes

When the job is running, `LSB_FRAMES` will be set to the relative frames with the format `LSB_FRAMES=start_number,end_number,step`.

From the *start_number*, *end_number*, and *step*, the frame job can know how many frames it will process.

Where defined

Set by `sbatchd`

Example

```
LSB_FRAMES=10,20,1
```

LSB_HMS_TIME_FORMAT

This parameter can be set from the command line or from `lsf.conf`.

See `LSB_HMS_TIME_FORMAT` in `lsf.conf`.

LSB_HOSTS

Syntax

`LSB_HOSTS="host_name..."`

Description

A list of hosts selected by LSF to run the job.

Notes

If a job is run on a single processor, the system sets `LSB_HOSTS` to the name of the host used.

For parallel jobs, the system sets `LSB_HOSTS` to the names of all the hosts used.

Where defined

Set by **sbatchd** when the job is executed. LSB_HOSTS is set only when the list of host names is less than 4096 bytes.

See also

LSB_MCPU_HOSTS

LSB_INTERACTIVE**Syntax**

LSB_INTERACTIVE=Y

Description

Indicates an interactive job. When you submit an interactive job using **bsub -I**, the system sets LSB_INTERACTIVE to Y.

Valid values

LSB_INTERACTIVE=Y (if the job is interactive)

Default

Not defined (if the job is not interactive)

Where defined

Set by sbatchd

LSB_JOB_AFF_REQ**Description**

Affinity preferences for the job as specified by the **bsub -jobaff** command option.

Where defined

From the command line.

LSB_JOB_INCLUDE_POSTPROC**Syntax**

LSB_JOB_INCLUDE_POSTPROC=Y | y | N | n

Description

Enables the post-execution processing of the job to be included as part of the job.

LSB_JOB_INCLUDE_POSTPROC in the user environment overrides the value of JOB_INCLUDE_POSTPROC in `lsb.params` and `lsb.applications`.

Default

Not defined

Environment variable reference

Where defined

From the command line.

LSB_JOBEXIT_INFO

Syntax

```
LSB_JOBEXIT_INFO="SIGNAL signal_value signal_name"
```

Description

Contains information about signal that caused a job to exit.

Applies to post-execution commands. Post-execution commands are set with POST_EXEC in `lsb.queues`.

When the post-execution command is run, the environment variable LSB_JOBEXIT_INFO is set if the job is signalled internally. If the job ends successfully, or the job is killed or signalled externally, LSB_JOBEXIT_INFO is not set.

Examples

```
LSB_JOBEXIT_INFO="SIGNAL -1 SIG_CHKPN" LSB_JOBEXIT_INFO="SIGNAL -14 SIG_TERM_USER"  
LSB_JOBEXIT_INFO="SIGNAL -23 SIG_KILL_REQUEUE"
```

Default

Not defined

Where defined

Set by sbatchd

LSB_JOBEXIT_STAT

Syntax

```
LSB_JOBEXIT_STAT=exit_status
```

Description

Indicates a job's exit status.

Applies to post-execution commands. Post-execution commands are set with POST_EXEC in `lsb.queues`.

When the post-execution command is run, the environment variable LSB_JOBEXIT_STAT is set to the exit status of the job. Refer to the man page for the `wait(2)` command for the format of this exit status.

The post-execution command is also run if a job is requeued because the job's execution environment fails to be set up, or if the job exits with one of the queue's REQUEUE_EXIT_VALUES. The LSB_JOBPEND environment variable is set if the job is requeued. If the job's execution environment could not be set up, LSB_JOBEXIT_STAT is set to 0.

Valid values

Any positive integer

Where defined

Set by sbatchd

LSB_JOBFILENAME

Syntax

LSB_JOBFILENAME=*file_name*

Description

The path to the batch executable job file that invokes the batch job. The batch executable job file is a **/bin/sh** script on UNIX systems or a **.BAT** command script on Windows systems.

LSB_JOBGROUP

Syntax

LSB_JOBGROUP=*job_group_name*

Description

The name of the job group associated with the job. When a job is dispatched, if it belongs to a job group, the runtime variable LSB_JOBGROUP is defined as its group. For example, if a dispatched job belongs to job group /X, LSB_JOBGROUP=/X.

Where defined

Set during job execution based on **bsub** options or the default job group defined in DEFAULT_JOBGROUP in `lsb.params` and the LSB_DEFAULT_JOBGROUP environment variable.

Default

Not defined

LSB_JOBID

Syntax

LSB_JOBID=*job_ID*

Description

The job ID assigned by **sbatchd**. This is the ID of the job assigned by LSF, as shown by **bjobs**.

Valid values

Any positive integer

Where defined

Set by **sbatchd**, defined by **mbatchd**

See also

LSB_REMOTEJID

LSB_JOBINDEX

Syntax

LSB_JOBINDEX=*index*

Description

Contains the job array index.

Valid values

Any integer greater than zero but less than the maximum job array size.

Notes

LSB_JOBINDEX is set when each job array element is dispatched. Its value corresponds to the job array index. LSB_JOBINDEX is set for all jobs. For non-array jobs, LSB_JOBINDEX is set to zero (0).

Where defined

Set during job execution based on **bsub** options.

Example

You can use LSB_JOBINDEX in a shell script to select the job command to be performed based on the job array index.

For example:

```
if [LSB_JOBINDEX -eq 1]; then cmd1 fi if [LSB_JOBINDEX -eq 2]; then cmd2 fi
```

See also

LSB_JOBINDEX_STEP, LSB_REMOTEINDEX

LSB_JOBINDEX_STEP

Syntax

LSB_JOBINDEX_STEP=*step*

Description

Step at which single elements of the job array are defined.

Valid values

Any integer greater than zero but less than the maximum job array size

Default

1

Notes

LSB_JOBINDEX_STEP is set when a job array is dispatched. Its value corresponds to the step of the job array index. This variable is set only for job arrays.

Where defined

Set during job execution based on **bsub** options.

Example

The following is an example of an array where a step of 2 is used:

```
array[1-10:2] elements:1 3 5 7 9
```

If this job array is dispatched, then `LSB_JOBINDEX_STEP=2`

See also

`LSB_JOBINDEX`

LSB_JOBNAME

Syntax

```
LSB_JOBNAME=job_name
```

Description

The name of the job defined by the user at submission time.

Default

The job's command line

Notes

The name of a job can be specified explicitly when you submit a job. The name does not have to be unique. If you do not specify a job name, the job name defaults to the actual batch command as specified on the **bsub** command line.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows.

Where defined

Set by `sbatchd`

Example

When you submit a job using the `-J` option of **bsub**, for example:

```
% bsub -J "myjob" job
```

`sbatchd` sets `LSB_JOBNAME` to the job name that you specified:

```
LSB_JOBNAME=myjob
```

LSB_JOBPEND

Description

Set if the job is requeued.

Where defined

Set by `sbatchd` for `POST_EXEC` only

See also

`LSB_JOBEXIT_STAT`, `REQUEUE_EXIT_VALUES`, `POST_EXEC`

LSB_JOBPGIDS

Description

A list of the current process group IDs of the job.

Where defined

The process group IDs are assigned by the operating system, and LSB_JOBPGIDS is set by `sbatchd`.

See also

LSB_JOBPIIDS

LSB_JOBPIIDS

Description

A list of the current process IDs of the job.

Where defined

The process IDs are assigned by the operating system, and LSB_JOBPIIDS is set by `sbatchd`.

See also

LSB_JOBPGIDS

LSB_MAILSIZE

Syntax

LSB_MAILSIZE=*value*

Description

Gives an estimate of the size of the batch job output when the output is sent by email. It is not necessary to configure LSB_MAILSIZE_LIMIT.

LSF sets LSB_MAILSIZE to the size in KB of the job output, allowing the custom mail program to intercept output that is larger than desired.

LSB_MAILSIZE is not recognized by the LSF default mail program. To prevent large job output files from interfering with your mail system, use LSB_MAILSIZE_LIMIT to explicitly set the maximum size in KB of the email containing the job information.

Valid values

A positive integer

If the output is being sent by email, LSB_MAILSIZE is set to the estimated mail size in kilobytes.

-1

If the output fails or cannot be read, LSB_MAILSIZE is set to -1 and the output is sent by email using LSB_MAILPROG if specified in `lsf.conf`.

Not defined

If you use the `-o` or `-e` options of `bsub`, the output is redirected to an output file. Because the output is not sent by email in this case, LSB_MAILSIZE is not used and LSB_MAILPROG is not called.

If the `-N` option is used with the `-o` option of `bsub`, LSB_MAILSIZE is not set.

Where defined

Set by sbatchd when the custom mail program specified by LSB_MAILPROG in `lsf.conf` is called.

LSB_MAX_NUM_PROCESSORS**Syntax**

`LSB_MAX_NUM_PROCESSORS=integer`

Description

The maximum number of processors requested when the job is submitted.

Where defined

Set during job execution based on **bsub** options. For example, for a job submitted with `-n 2,4`, the maximum number of processors requested is 4.

LSB_MCPU_HOSTS**Syntax**

`LSB_MCPU_HOSTS="host_nameA num_processors1 host_nameB num_processors2..."`

Description

Contains a list of the hosts and the number of CPUs used to run a job.

Valid values

`num_processors1`, `num_processors2`,... refer to the number of CPUs used on `host_nameA`, `host_nameB`,..., respectively

Default

Not defined

Notes

The environment variables `LSB_HOSTS` and `LSB_MCPU_HOSTS` both contain the same information, but the information is presented in different formats. `LSB_MCPU_HOSTS` uses a shorter format than `LSB_HOSTS`. As a general rule, sbatchd sets both these variables. However, for some parallel jobs, `LSB_HOSTS` is not set.

For parallel jobs, several CPUs are used, and the length of `LSB_HOSTS` can become very long. sbatchd needs to spend a lot of time parsing the string. If the size of `LSB_HOSTS` exceeds 4096 bytes, `LSB_HOSTS` is ignored, and sbatchd sets only `LSB_MCPU_HOSTS`.

To verify the hosts and CPUs used for your dispatched job, check the value of `LSB_HOSTS` for single CPU jobs, and check the value of `LSB_MCPU_HOSTS` for parallel jobs.

Where defined

Set by sbatchd before starting a job on the execution host

Example

When the you submit a job with the `-m` and `-n` options of **bsub**, for example,

```
% bsub -m "hostA hostB" -n 6 job
```

Environment variable reference

sbatchd sets the environment variables LSB_HOSTS and LSB_MCPU_HOSTS as follows:

```
LSB_HOSTS= "hostA hostA hostA hostB hostB hostB"  
LSB_MCPU_HOSTS="hostA 3 hostB 3"
```

Both variables are set in order to maintain compatibility with earlier versions.

See also

LSB_HOSTS

LSB_NQS_PORT

This parameter can be defined in `lsf.conf` or in the services database such as `/etc/services`.

See `LSB_NUM_NIOS_CALLBACK_THREADS` in `lsf.conf` for more details.

LSB_NTRIES

Syntax

`LSB_NTRIES=integer`

Description

The number of times that LSF libraries attempt to contact mbatchd or perform a concurrent jobs query.

For example, if this parameter is not defined, when you type **bjobs**, LSF keeps displaying "batch system not responding" if mbatchd cannot be contacted or if the number of pending jobs exceeds `MAX_PEND_JOBS` specified in `lsb.params` or `lsb.users`.

If this parameter is set to a value, LSF only attempts to contact mbatchd the defined number of times and then quits. LSF will wait for a period of time equal to `SUB_TRY_INTERVAL` specified in `lsb.params` before attempting to contact mbatchd again.

Valid values

Any positive integer

Default

`INFINIT_INT` (The default is to continue the attempts to contact mbatchd)

LSB_OLD_JOBID

Syntax

`LSB_OLD_JOBID=job_ID`

Description

The job ID of a job at the time it was checkpointed.

When a job is restarted, it is assigned a new job ID and `LSB_JOBID` is replaced with the new job ID. `LSB_OLD_JOBID` identifies the original ID of a job before it is restarted.

Valid values

Any positive integer

Where defined

Set by sbatchd, defined by mbatchd

See also

LSB_JOBID

LSB_OUTDIR

String containing the full path to the output directory specified by the **bsub -outdir** option when a job is submitted. If an output directory wasn't specified, this variable contains the path to the job submission directory.

The LSF data management **bstage out** command uses this directory as the default destination for jobs being transferred out.

LSB_OUTPUT_TARGETFAILED**Syntax**

LSB_OUTPUT_TARGETFAILED=Y

Description

Indicates that LSF cannot access the output file specified for a job submitted the **bsub -o** option.

Valid values

Set to Y if the output file cannot be accessed; otherwise, it is not defined.

Where defined

Set by sbatchd during job execution

LSB_PROJECT_NAME**Syntax**

LSB_PROJECT_NAME=*project_name*

Description

To enable the use of project names for accounting purposes inside third party tools that launch jobs under LSF using environment variables, the environment variable, **LSB_PROJECT_NAME** is used.

When **bsub -P** is used to specify a project name, **LSB_PROJECT_NAME** is set with the project name.

Default

Not defined

Notes

Project names can be up to 511 characters long.

There are three ways to set the project name for a job. The order of priority for each is:

1. If you submit a job with the **-P** option of **bsub**, the job belongs to the project specified through the **-P** option (sets **LSB_PROJECT_NAME**).
2. If you submit a job without the **-P** option of **bsub**, but you defined **LSB_DEFAULTPROJECT**, then the job belongs to the project specified in **LSB_DEFAULTPROJECT**.

Environment variable reference

3. If the LSF administrator defines **DEFAULT_PROJECT** in the `lsb.params` configuration file, the system uses this as the default project name (unless **LSB_DEFAULTPROJECT** is set or **bsub -P** is used).

Where defined

From the command line, or through the `-P` option of **bsub**

See also

LSB_DEFAULTPROJECT environment variable, **DEFAULT_PROJECT** in `lsb.params`, the `-P` option of **bsub**

LSB_QUEUE

Syntax

`LSB_QUEUE=queue_name`

Description

The name of the queue from which the job is dispatched.

Where defined

Set by **sbatchd**

LSB_RANK_HOSTFILE

Syntax

`LSB_RANK_HOSTFILE=file_path`

Description

Path to the OpenMPI host rank file specified by the `DJOB_ENV_SCRIPT` option in `lsb.applications`.

If `DJOB_ENV_SCRIPT=openmpi_rankfile.sh` is set in `lsb.applications`, LSF creates a host rank file and sets the environment variable `LSB_RANK_HOSTFILE`.

On the first execution host, LSF **sbatchd** will create a host rank file per job under the same location as `$LSB_DJOB_HOSTFILE`.

Default

Not defined

Where defined

Set during job execution.

LSB_REMOTEINDEX

Syntax

`LSB_REMOTEINDEX=index`

Description

The job array index of a remote MultiCluster job. `LSB_REMOTEINDEX` is set only if the job is an element of a job array.

Valid values

Any integer greater than zero, but less than the maximum job array size

Where defined

Set by `sbatchd`

See also

LSB_JOBINDEX, MAX_JOB_ARRAY_SIZE in `lsb.params`

LSB_REMOTEJID**Syntax**

LSB_REMOTEJID=*job_ID*

Description

The job ID of a remote MultiCluster job.

Where defined

Set by `sbatchd`, defined by `mbatchd`

See also

LSB_JOBID

LSB_JOB_REPORT_MAIL**Syntax**

LSB_JOB_REPORT_MAIL=Y | N | ERROR

Description

End users can set the value for this environment variable to N in the job submission environment to disable email notification for only that particular job and not email notification for all jobs. In this case, there is no need to restart **sbatchd**. Mail generated from **mbatchd** is sent as usual. If

LSB_JOB_REPORT_MAIL=Y, job-related email is sent.

End users can set the value for this environment variable to ERROR in the job submission environment to send mail only when the job exits (that is, when the job is under `Exit` status). This parameter only affects email sent by **sbatchd**. This ensures that an email notification is only sent on a job error.

Where defined

From the command line.

LSB_RESTART**Syntax**

LSB_RESTART=Y

Description

Indicates that a job has been restarted or migrated.

Environment variable reference

Valid values

Set to Y if a checkpointable job has been restarted or migrated; otherwise, it is undefined.

Notes

If a checkpointable job is submitted with the `-k` option of the `bsub` command, the `LSB_RESTART` variable is set to Y when the job is restarted. If the `bmig` command is used to migrate a job, the `LSB_RESTART` variable is set to Y when the migrated job is restarted.

If the job is not a restarted job, then the `LSB_RESTART` variable is not set.

Where defined

Set by `sbatchd` during job execution

See also

`LSB_RESTART_PGID`, `LSB_RESTART_PID`

LSB_RESTART_PGID

Syntax

`LSB_RESTART_PGID=pgid`

Description

The process group ID of the checkpointed job when the job is restarted.

Notes

When a checkpointed job is restarted, the operating system assigns a new group process ID to the job. LSF sets `LSB_RESTART_PGID` to the new group process ID.

Where defined

Set during restart of a checkpointed job.

See also

`LSB_RESTART_PID`, `LSB_RESTART`

LSB_RESTART_PID

Syntax

`LSB_RESTART_PID=pid`

Description

The process ID of the checkpointed job when the job is restarted.

Notes

When a checkpointed job is restarted, the operating system assigns a new process ID to the job. LSF sets `LSB_RESTART_PID` to the new process ID.

Where defined

Defined during restart of a checkpointed job

See also

LSB_RESTART_PGID, LSB_RESTART

LSB_RTASK_GONE_ACTION**Syntax**

LSB_RTASK_GONE_ACTION=*task_action* ...

Description

Defines the actions LSF should take if it detects that a remote task of a parallel job is gone. Where *task_action* is:

IGNORE_TASKCRASH

A remote task crashes. The job RES does nothing.

KILLJOB_TASKDONE

A remote task exits with zero value. The job RES notifies the task RES to terminate all tasks in the job.

KILLJOB_TASKEXIT

A remote task exits with non-zero value. The job RES notifies the task RES to terminate all tasks in the job.

Where defined

Set by the system based on the value of the parameter RTASK_GONE_ACTION in `lsb.applications` when running **bsub -app** for the specified application

See also

RTASK_GONE_ACTION in `lsb.applications`

LSB_SUB_ADDITIONAL**Usage**

LSB_SUB_ADDITIONAL=*esub executables*

Description

String that contains the application name or names of the **esub** executables requested by the user. This is the only option that an **esub** cannot change or add at job submission.

Default

Not defined

Where defined

Set by LSF on the submission host before running **esub**

LSB_SUB_CLUSTER**Description**

Name of submission cluster (MultiCluster only)

Environment variable reference

Where defined

Set on the submission environment and passed to the execution cluster environment. The parameter will ONLY be valid in Multi Cluster environment. For jobs on a local cluster, the parameter is not set when using any daemon wrappers such as job starter, post-, pre- or eexec scripts.

LSB_SUB_COMMANDNAME

Description

The job command line.

If set, enables **esub** to use the variable **LSB_SUB_COMMAND_LINE** in the **esub** job parameter file specified by the **\$LSB_SUB_PARM_FILE** environment variable. The **LSB_SUB_COMMAND_LINE** variable carries the value of the **bsub** command argument, and is used when **esub** runs.

Where defined

Set by **esub** before a job is submitted.

LSB_SUB_COMMAND_LINE

Description

The job command line.

The job command line can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows.

Where defined

Set by **esub** before a job is submitted.

LSB_SUB_EXTSCHED_PARAM

Description

Value of external scheduling options specified by **bsub -extsched**, or queue-level MANDATORY_EXTSCHED or DEFAULT_EXTSCHED.

Where defined

Set by **esub** before a job is submitted.

LSB_SUB_JOB_ACTION_WARNING_TIME

Description

Value of job warning time period specified by **bsub -wt**.

Where defined

Set by **esub** before a job is submitted.

LSB_SUB_JOB_ERR

Syntax

LSB_SUB_JOB_ERR=*lsb_err_number*

Description

The error number for an externally-submitted job defined by **mbatchd** if the job submission failed. This is available to the external post-submission scripts (**epsub**) to determine the reason for the job submission failure.

If the job submission is successful, this value is **LSB_NO_ERROR** (or 0).

Valid values

Any non-negative integer.

A value of **LSB_NO_ERROR** (or 0) indicates that the job submission is successful.

Where defined

Set by **bsub** or **bmod**, defined by **mbatchd**.

See also

LSB_NO_ERROR

LSB_SUB_JOB_ID**Syntax**

`LSB_SUB_JOB_ID=job_ID | -1`

Description

The job ID assigned by **mbatchd**. This is available to the external post-submission scripts (**epsub**).

Valid values

Any positive integer, or -1 to indicate that no job ID was received from **mbatchd**.

Where defined

Set by **bsub** or **bmod**, defined by **mbatchd**.

See also

LSB_JOBID

LSB_SUB_JOB_QUEUE**Syntax**

`LSB_SUB_JOB_QUEUE=queue_name`

Description

The name of the final queue from which the job is dispatched, which includes any queue modifications made by **esub**. This is available to the external post-submission scripts (**epsub**).

Where defined

Set by **bsub** or **bmod**, defined by **mbatchd**.

LSB_SUB_JOB_WARNING_ACTION

Description

Value of job warning action specified by **bsub -wa**.

Where defined

Set by **esub** before a job is submitted.

LSB_SUB_MEM_SWAP_HOST_LIMIT

Syntax

LSB_SUB_MEM_SWAP_HOST_LIMIT=Y | SUB_RESET

Description

Controls per-job host-based memory and swap limit enforcement on hosts that support Linux cgroups. **LSB_RESOURCE_ENFORCE="memory"** must be specified in `lsf.conf` and a memory or swap limit must be specified for the job for host-based memory and swap limit enforcement to take effect.

Where defined

Set by **esub** at job submission.

LSB_SUB_PARM_FILE

Syntax

LSB_SUB_PARM_FILE=*file_name*

Description

Points to a temporary file that LSF uses to store the **bsub** options entered in the command line. An **esub** reads this file at job submission and either accepts the values, changes the values, or rejects the job. Job submission options are stored as name-value pairs on separate lines in the format `option_name=value`. A typical use of this file is to control job submission options.

Where defined

Set by LSF on the submission host before running **esub**. Not defined when **lsrun** or **lsgrun** are used for interactive remote execution.

LSB_SUB_RES_REQ

Syntax

LSB_SUB_RES_REQ=*res_req_string*

Description

The job level resource requirement for the job submission. If the job level resource requirement was changed by **bmod -R** for a running job, the changed resource requirement is not available via this environment variable.

Where defined

Set by **bsub** at job submission.

LSB_SUB_USER

Syntax

LSB_SUB_USER=*user_name*

Description

The user name for the job submission.

Where defined

Set by **bsub** at job submission.

LSB_SUCCESS_EXIT_VALUES

Syntax

LSB_SUCCESS_EXIT_VALUES=[*exit_code ...*]

Description

Specifies the exit values that indicate successful execution for applications that successfully exit with non-zero values. Use spaces to separate multiple exit codes. *exit_code* should be the value between 0 and 255.

User-defined LSB_SUCCESS_EXIT_VALUES overrides application profile level specification of SUCCESS_EXIT_VALUES in `lsb.applications`.

LSB_SUSP_REASONS

Syntax

LSB_SUSP_REASONS=*integer*

Description

An integer representing suspend reasons. Suspend reasons are defined in `lsbatch.h`.

This parameter is set when a job goes to system-suspended (SSUSP) or user-suspended status (USUSP). It indicates the exact reason why the job was suspended.

To determine the exact reason, you can test the value of LSB_SUSP_REASONS against the symbols defined in `lsbatch.h`.

Where defined

Set during job execution

See also

LSB_SUSP_SUBREASONS

LSB_SUSP_SUBREASONS

Syntax

LSB_SUSP_SUBREASONS=*integer*

Environment variable reference

Description

An integer representing the load index that caused a job to be suspended.

When the suspending reason `SUSP_LOAD_REASON` (suspended by load) is set in `LSB_SUSP_REASONS`, `LSB_SUSP_SUBREASONS` set to one of the load index values defined in `lsf.h`.

Use `LSB_SUSP_REASONS` and `LSB_SUSP_SUBREASONS` together in your custom job control to determine the exact load threshold that caused a job to be suspended.

Load index values are defined in `lsf.h`.

Load Index	Value
R15S	0
R1M	1
R15M	2
UT	3
PG	4
IO	5
LS	6
IT	7
TMP	8
SWP	9
MEM	10

Default

Not defined

Where defined

Set during job execution

See also

`LSB_SUSP_REASONS`

LSB_SWITCH_ABORT_VALUE

Usage

`LSB_SWITCH_ABORT_VALUE=exit_code`

Description

Indicates the value that **eswitch** should exit with if LSF is to reject the job switch request.

Notes

If **eswitch** exits with the value **LSB_SWITCH_ABORT_VALUE**, LSF will reject the job submission. For example, to reject a job in a Bourne shell **eswitch**:

```
exit $LSB_SWITCH_ABORT_VALUE
```

Default

Not defined

Where defined

Set by LSF on the submission host before running **eswitch**

See also

LSB_SWITCH_PARM_FILE

LSB_SWITCH_MODIFY_FILE**Description**

Points to the file that the **eswitch** uses to modify the **bswitch** target queue values.

When the **eswitch** runs, LSF checks the specified file and applies changes so that the job switches to the revised queue. The job switch request is rejected when **eswitch** encounters any parameters in the **LSB_SWITCH_MODIFY_FILE** file other than **LSB_SWITCH_QUEUE**.

Where defined

Set by **eswitch** before a job is switched.

LSB_SWITCH_PARM_FILE**Syntax**

```
LSB_SWITCH_PARM_FILE=file_name
```

Description

Points to a temporary file that LSF uses to store the **bswitch** options entered in the command line. An **eswitch** reads this file at the job **bswitch** level and either accepts or changes the values. Job switch options are stored as name-value pairs on separate lines in the format `option_name=value`. A typical use of this file is to control job switch options.

Where defined

Set by LSF on the submission host before running the **eswitch**.

LSB_TASK_GEOMETRY**Usage**

```
setenv LSB_TASK_GEOMETRY "{(task_ID,...) ...}"
```

Description

Specifies task geometry for your jobs. **LSB_TASK_GEOMETRY** overrides any process group or command file placement options. This environment variable is checked for all parallel jobs. If set, users submit a

Environment variable reference

parallel job (a job that requests more than 1 slot) and LSF attempts to shape **LSB_MCPU_HOSTS** accordingly.

LSB_TASK_GEOMETRY replaces **LSB_PJL_TASK_GEOMETRY**, which is kept for compatibility with earlier versions

Where defined

From the command line.

LSB_UNIXGROUP

Description

Specifies the UNIX user group of the submitting user.

Notes

This variable is useful if you want pre- or post-execution processing to use the user group of the user who submitted the job, and not **sys(1)**.

Where defined

Set during job execution

LSB_USER_BIND_CPU_LIST

The binding requested at job submission takes effect when **LSF_BIND_JOB=USER_CPU_LIST** in `lsf.conf` or **BIND_JOB=USER_CPU_LIST** in an application profile in `lsb.applications`. LSF makes sure that the value is in the correct format, but does not check that the value is valid for the execution hosts.

The correct format is a list which may contain multiple items, separated by comma, and ranges. For example: 0,5,7,9-11.

LSB_USER_BIND_JOB

The binding requested at job submission takes effect when **LSF_BIND_JOB=USER** in `lsf.conf` or **BIND_JOB=USER** in an application profile in `lsb.applications`. This value must be one of Y, N, NONE, BALANCE, PACK, or ANY. Any other value is treated as ANY.

LSF_AC_JOB_NOTIFICATION

Syntax

```
LSF_AC_JOB_NOTIFICATION="[exit ][done ][start ][suspend]"
```

Description

Requests that the user be notified when the job reaches any of the specified states.

Use a space to separate multiple job states. The notification request is logged in **JOB_NEW** events.

Where defined

Set before job submission

LSF_CMD_LOGDIR

This parameter can be set from the command line or from `lsf.conf`.

See **LSF_CMD_LOGDIR** in `lsf.conf`.

LSF_DEBUG_CMD

This parameter can be set from the command line or from `lsf.conf`.

See `LSB_DEBUG_MBD` in `lsf.conf`.

LSF_DEBUG_LIM

This parameter can be set from the command line or from `lsf.conf`.

See `LSF_DEBUG_LIM` in `lsf.conf`.

LSF_DEBUG_RES

This parameter can be set from the command line or from `lsf.conf`.

See `LSF_DEBUG_RES` in `lsf.conf`.

LSF_EAUTH_AUX_DATA**Syntax**

`LSF_EAUTH_AUX_DATA=path/file_name`

Description

Used in conjunction with LSF daemon authentication, specifies the full path to the temporary file on the local file system that stores auxiliary authentication information (such as credentials required by a remote host for use during job execution). Provides a way for `eauth -c`, `mbatchd`, and `sbatchd` to communicate the location of auxiliary authentication data. Set internally by the LSF libraries in the context of `eauth`.

For Kerberos authentication, used for forwarding credentials to the execution host.

LSF_EAUTH_AUX_PASS**Syntax**

`LSF_EAUTH_AUX_PASS=yes`

Description

Enables forwarding of credentials from a submission host to an execution host when daemon authentication is enabled. `LSF_EAUTH_AUX_PASS=yes` indicates that a credential can be added to the execution context of a job. Set to `yes` by `bsub` during job submission or by `bmod` during job modification so that `eauth -c` can forward credentials.

LSF_EAUTH_CLIENT**Syntax**

`LSF_EAUTH_CLIENT=mbatchd | sbatchd | pam | res | user`

Description

Used with LSF daemon authentication, specifies the LSF daemon, command, or user that invokes `eauth -c`. Used when writing a customized `eauth` executable to set the context for the call to `eauth`. Set internally by the LSF libraries or by the LSF daemon, command, or user calling `eauth -c`.

LSF_EAUTH_SERVER**Syntax**

`LSF_EAUTH_SERVER=mbatchd | sbatchd | pam | res`

Environment variable reference

Description

Used with LSF daemon authentication, specifies the daemon that invokes `eauth -s`. Used when writing a customized `eauth` executable to set the context for the call to `eauth`. Set internally by the LSF libraries or by the LSF daemon calling `eauth -s`.

LSF_EAUTH_UID

Syntax

`LSF_EAUTH_UID=user_ID`

Description

Specifies the user account under which `eauth -s` runs. Set by the LSF daemon that executes `eauth`.

LSF_EXECUTE_DOMAIN

Syntax

`LSF_EXECUTE_DOMAIN=domain_namesetenv LSF_EXECUTE_DOMAIN domain_name`

Description

If UNIX/Windows user account mapping is enabled, specifies the preferred Windows execution domain for a job submitted by a UNIX user. The execution domain must be one of the domains listed in `LSF_USER_DOMAIN`.

`LSF_EXECUTE_DOMAIN` is defined in the user environment (`.cshrc` or `.profile`) or from the command line. Specify only one domain.

Use this parameter in conjunction with the `bsub`, `lrun`, and `lsgun` commands to bypass the order of the domains listed in `LSF_USER_DOMAIN` and run the job using the specified domain. If you do not have a Windows user account in the execution domain, LSF tries to run the job using one of the other domains defined by `LSF_USER_DOMAIN`. Once you submit a job with an execution domain defined, you cannot change the execution domain for that particular job.

LSF_INTERACTIVE_STDERR

This parameter can be defined in `lsf.conf`.

See `LSF_INTERACTIVE_STDERR` in `lsf.conf` for more details.

LSF_INVOKE_CMD

Syntax

`LSF_INVOKE_CMD=invoking_command_name`

Description

Indicates the name of the last LSF command that invoked an external executable (for example, `esub` or `eexec`).

External executables get called by different LSF commands, such as `bsub`, `bmod`, or `lrun`.

Default

Not defined

Where defined

Set internally within by LSF

LSF_JOB_STARTER

Syntax

LSF_JOB_STARTER=*binary*

Description

Specifies an executable program that has the actual job as an argument.

Default

Not defined

Interactive Jobs

If you want to run an interactive job that requires some preliminary setup, LSF provides a job starter function at the command level. A command-level job starter allows you to specify an executable file that will run prior to the actual job, doing any necessary setup and running the job when the setup is complete.

If LSF_JOB_STARTER is properly defined, RES will invoke the job starter (rather than the job itself), supplying your commands as arguments.

Batch Jobs

A job starter can also be defined at the queue level using the JOB_STARTER parameter, although this can only be done by the LSF administrator.

Where defined

From the command line

Example: UNIX

The job starter is invoked from within a Bourne shell, making the command-line equivalent:

```
/bin/sh -c "$LSF_JOB_STARTER command [argument...]"
```

where *command* [*argument...*] are the command line arguments you specified in **lsrun**, **lsgrun**, or **ch**.

If you define LSF_JOB_STARTER as follows:

```
setenv LSF_JOB_STARTER "/bin/csh -c"
```

and run a simple C-shell job:

```
lsrun "'a.out; echo hi&apos;"
```

The following will be invoked to correctly start the job:

```
/bin/sh -c "/bin/csh -c &apos;a.out; echo hi&apos;"
```

Example: Windows

RES runs the job starter, passing it your commands as arguments:

```
LSF_JOB_STARTER command [argument...]
```

If you define LSF_JOB_STARTER as follows:

```
set LSF_JOB_STARTER=C:\cmd.exe /C
```

Environment variable reference

and run a simple DOS shell job:

```
C:\> lsrun dir /p
```

then the following will be invoked to correctly start the job:

```
C:\cmd.exe /C dir /p
```

See also

JOB_STARTER in `lsb.queues`

LSF_LD_LIBRARY_PATH

Description

When `LSF_LD_SECURITY=Y` in `lsf.conf`, contains the value of the `LD_LIBRARY_PATH` environment variable, which is removed from the job environment during job initialization to ensure enhanced security against users obtaining root privileges. `LSF_LD_LIBRARY_PATH` allows the `LD_LIBRARY_PATH` environment variable to be put back before the job runs.

Where defined

For jobs submitted using `bsub -Is` or `bsub -Ip` only.

See also

`LSF_LD_PRELOAD`, `LSF_LD_SECURITY` in `lsf.conf`

LSF_LD_PRELOAD

Description

When `LSF_LD_SECURITY=Y` in `lsf.conf`, contains the value of the `LD_PRELOAD` environment variable, which is removed from the job environment during job initialization to ensure enhanced security against users obtaining root privileges. `LSF_LD_PRELOAD` allows the `LD_PRELOAD` environment variable to be put back before the job runs.

Where defined

For jobs submitted using `bsub -Is` or `bsub -Ip` only.

See also

`LSF_LD_LIBRARY_PATH`, `LSF_LD_SECURITY` in `lsf.conf`

LSF_LIM_API_NTRIES

Syntax

`LSF_LIM_API_NTRIES=integer`

Description

Defines the number of times LSF commands will retry to communicate with the LIM API when LIM is not available. `LSF_LIM_API_NTRIES` is ignored by LSF and EGO daemons and EGO commands. The `LSF_LIM_API_NTRIES` environment variable overrides the value of `LSF_LIM_API_NTRIES` in `lsf.conf`.

Valid values

1 to 65535

Where defined

From the command line or from `lsf.conf`

Default

Not defined. If not defined in `lsf.conf`, LIM API exits without retrying.

LSF_LIM_DEBUG

This parameter can be set from the command line or from `lsf.conf`.

See `LSF_LIM_DEBUG` in `lsf.conf`.

LSF_LOGDIR

This parameter can be set from the command line or from `lsf.conf`.

See `LSF_LOGDIR` in `lsf.conf`.

LSF_LSLOAD_FORMAT

This parameter can be set from the command line or from `lsf.conf`.

See `LSF_LSLOAD_FORMAT` in `lsf.conf`.

LSF_LSHOSTS_FORMAT

This parameter can be set from the command line or from `lsf.conf`.

See `LSF_LSHOSTS_FORMAT` in `lsf.conf`.

LSF_MASTER**Description**

Set by the LIM to identify the master host. The value is Y on the master host and N on all other hosts. An **elim** executable can use this parameter to check the host on which the **elim** is currently running.

Used when the external load indices feature is enabled.

When defined

Set by the LIM when it starts the master external load information manager (MELIM).

See also

`LSF_RESOURCES`

LSF_NIOS_DEBUG

This parameter can be set from the command line or from `lsf.conf`.

See `LSF_NIOS_DEBUG` in `lsf.conf`.

LSF_NIOS_ERR_LOGDIR

This parameter can be set from the command line or from `lsf.conf`.

See `LSF_NIOS_ERR_LOGDIR` in `lsf.conf`.

LSF_NIOS_DIE_CMD

Syntax

LSF_NIOS_DIE_CMD=*command*

Description

If set, the command defined by the **LSF_NIOS_DIE_CMD** environment variable is executed for a pending job before NIOS exits with a failure.

You can use the %J special character to specify the command. %J is replaced by the job ID.

Default

bkill %J

Where defined

From the command line

LSF_NIOS_IGNORE_SIGWINDOW

Syntax

LSF_NIOS_IGNORE_SIGWINDOW=*any_value*

Description

If defined, the NIOS will ignore the SIGWINDOW signal.

Default

Not defined

Notes

When the signal SIGWINDOW is defined, some tasks appear to die when they receive the SIGWINDOW while doing I/O. By defining LSF_NIOS_IGNORE_SIGWINDOW, these tasks are given the chance to ignore the signal.

Where defined

From the command line

LSF_NIOS_PEND_TIMEOUT

Syntax

LSF_NIOS_PEND_TIMEOUT=*minutes*

Description

Applies only to interactive batch jobs.

Maximum amount of time that an interactive batch job can remain pending.

If this parameter is defined, and an interactive batch job is pending for longer than the specified time, LSF executes the command as specified by the **LSF_NIOS_DIE_CMD** environment variable and returns the output of that command.

Valid values

Any integer greater than zero

Default

Not defined

LSF_NIOS_PORT_RANGE**Syntax**

`LSF_NIOS_PORT_RANGE=min_port_number-max_port_number`

Description

Defines a range of listening ports for NIOS to use.

Example

```
LSF_NIOS_PORT_RANGE=5000-6000
```

Default

Not defined. LSF randomly assigns a NIOS port number.

LSF_RESOURCES**Syntax**

`LSF_RESOURCES=dynamic_external_resource_name...`

Description

Space-separated list of dynamic external resources. When the LIM starts a master external load information manager (MELIM) on a host, the LIM checks the resource mapping defined in the ResourceMap section of `lsf.cluster.cluster_name`. Based on the mapping (default, all, or a host list), the LIM sets LSF_RESOURCES to the list of resources expected on the host and passes the information to the MELIM.

Used when the external load indices feature is enabled.

When defined

Set by the MELIM on the host when the MELIM invokes the **elim** executable.

See also

LSF_MASTER

LSF_TS_LOGON_TIME**Syntax**

`LSF_TS_LOGON_TIME=milliseconds`

Description

Specifies the time to create a Windows Terminal Service session. Configure LSF_TS_LOGON_TIME according to the load on your network environment.

Environment variable reference

The default, 30000 milliseconds, is suitable for most environments. If you set LSF_TS_LOGON_TIME too small, the LSF tries multiple times before it succeeds in making a TS session with the TS server, which can cause the job wait a long time before it runs. For a congested network, set LSF_TS_LOGON_TIME=1000000.

Where defined

From the command line

Default

30000 milliseconds

LSF_USE_HOSTEQUIV

Syntax

LSF_USE_HOSTEQUIV=y | Y

Description

Used for authentication purposes. If LSF_USE_HOSTEQUIV is defined, RES and mbatchd call the `ruserok(3)` function to decide if a user is allowed to run remote jobs. LSF trusts all hosts configured in the LSF cluster that are defined in `hosts.equiv`, or in `.rhosts` in the user's home directory.

The `ruserok(3)` function checks in the `/etc/hosts.equiv` file and the user's `$HOME/.rhosts` file to decide if the user has permission to execute remote jobs.

If LSF_USE_HOSTEQUIV is not defined, all normal users in the cluster can execute remote jobs on any host.

If LSF_ROOT_REX is set, root can also execute remote jobs with the same permission test as for normal users.

Default

Not defined

See also

LSF_ROOT_REX and LSF_AUTH in `lsf.conf`

LSF_USER_DOMAIN

Syntax

LSF_USER_DOMAIN=*domain_name* / .

Description

Set during LSF installation or setup. If you modify this parameter in an existing cluster, you probably have to modify passwords and configuration files also.

Windows or mixed UNIX-Windows clusters only.

Enables default user mapping, and specifies the LSF user domain. The period (.) specifies local accounts, not domain accounts.

- A user name specified without a domain is interpreted (on a Windows host) as belonging to the LSF user domain
- A user name specified with the domain name of the LSF user domain is not valid

- In a mixed cluster, this parameter defines a 2-way, 1:1 user map between UNIX user accounts and Windows user accounts belonging to the specified domain, as long as the accounts have the same user name. This means jobs submitted by the Windows user account can run on a UNIX host, and jobs submitted by the UNIX account can run on any Windows host that is available to the Windows user account.

If this parameter is not defined, the default user mapping is not enabled. You can still configure user mapping at the user or system level. User account mapping is required to run cross-platform jobs in a UNIX-Windows mixed cluster.

Where defined

lsf.conf

Default

- If you upgrade from LSF 4.0.1 or earlier, the default is the existing LSF user domain.
- For a new, Windows-only cluster, this parameter is not defined (no LSF user domain, no default user mapping).
- For a new, mixed UNIX-Windows cluster, the default is the domain that the Windows installation account belongs to. This can be modified during LSF installation.

NOCHECKVIEW_POSTEXEC

LSF integration for Rational ClearCase. Setting the NOCHECKVIEW_POSTEXEC environment variable to 1 configures the daemon wrapper to skip running the checkView function during the post-execution phase of a job.

Syntax

NOCHECKVIEW_POSTEXEC=1

Description

Use one of the following shell commands to set the NOCHECKVIEW_POSTEXEC environment variable:

- In **cs**h or **tc**sh:

```
setenv NOCHECKVIEW_POSTEXEC 1
```

- In **sh**, **k**sh, or **ba**sh:

```
export NOCHECKVIEW_POSTEXEC=1
```

OMP_NUM_THREADS

Syntax

OMP_NUM_THREADS=*num_cpus*

Description

LSF will set the environment variable OMP_NUM_THREADS to the total number of CPUs allocated to each task if the job requests more than one per task (that is, the job is a multithreaded job).

If you set OMP_NUM_THREADS before submitting your job, LSF does not change it, but passes it to each task.

If OMP_NUM_THREADS is not set in the job submission environment, and the job has an affinity resource request, and *number_pu * number_sub_tasks* > 1, then OMP_NUM_THREADS is set to *number_pu * number_sub_tasks*. If *number_pu * number_sub_tasks* ≤ 1 OMP_NUM_THREADS is not set.

Environment variable reference

For example: for a job submitted with `bsub -n 2 -R "affinity[core(3)*4]"`, `OMP_NUM_THREADS` will be set to $3*4=12$

If the job has no affinity resource request, the openmp **esub** is specified (`bsub -a openmp`), `OMP_NUM_THREADS` is set according to the `span[]` resource requirements option:

- If `-R "span[hosts=x]"`, then `OMP_NUM_THREADS` is set
- If `-R "span[ptile=y]"`, then `OMP_NUM_THREADS` is not set

For example, for a job submitted with `bsub -a openmp -n 4 -R "span[ptile=2]"`, `OMP_NUM_THREADS` will be set to 2.

If the job does not specify the openmp**esub**, `OMP_NUM_THREADS` is not set.

Default

Not defined

Where defined

Set during job execution.

RM_CPUTASKn

Syntax

`RM_CPUTASKn=cpu_list`

Description

The `RM_CPUTASKn` environment variable is the IBM Parallel Environment equivalent of `LSB_BIND_CPU_LIST`. Each `RM_CPUTASK` represents the CPU allocation for one task, as opposed to having a single variable which amalgamates them all. For example a job with 6 tasks running on 4 hosts could have the following values for `RM_CPUTASK` (the example corresponds to the tasks specified in the example for `LSB_AFFINITY_HOSTFILE`):

On Host1:

```
RM_CPUTASK1=0,1,2,3
RM_CPUTASK2=4,5,6,7
```

On Host2:

```
RM_CPUTASK1=0,1,2,3
RM_CPUTASK2=4,5,6,7
```

On Host3:

```
RM_CPUTASK1=0,1,2,3
```

On Host4:

```
RM_CPUTASK1=0,1,2,3
```

Default

Not defined

Where defined

Set during job execution.

RM_MEM_AFFINITY

Syntax

RM_MEM_AFFINITY=Yes | No

Description

The RM_MEM_AFFINITY environment variable informs the IBM Parallel Environment memory binding policy. If a local-only policy (membind=localonly affinity resource requirement parameter) has been set, RM_MEM_AFFINITY will be set to Yes. If local preference policy (membind=localprefer affinity resource requirement parameter) has been set, RM_MEM_AFFINITY will be set to No.

Note: IBM Parallel Environment does not distinguish between a local-prefer memory policy and no policy at all.

Default

Not defined

Where defined

Set during job execution.

TASKMAN_EXEC_RUSAGE

Syntax

TASKMAN_EXEC_RUSAGE="*resource_name1 resource_value1 resource_name2 resource_value2...*"

Description

Indicates to **taskman** which rusage string is satisfied to permit the task to run. This environment variable is necessary because the OR (|) operator specifies alternative rusage strings for running tasks that are launched inside LSF jobs.

Valid values

resource_value1, resource_value2, . . . refer to the resource values on *resource_name1, resource_name2, . . .* respectively.

Default

Not defined

Where defined

Set by LSF after reserving a resource for the job.

