IBM LoadLeveler for AIX

IBM

IBM LoadLeveler for AIX

IBM

Using and Administering

Version 2 Release 1

SA22-7311-00

llclass - Query Class Information . . . . . .

Notices

References in this publication to IBM* products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Who Should Use This Book

Related Information

What's New in Version 2

The following is a list of new functions added for this release.

Multiple User Space Tasks per Adapter

LoadLeveler now supports running multiple user space tasks on an SP switch

New Scheduling Algorithm

LoadLeveler now includes an additional scheduling algorithm, the Backfill scheduler. This scheduler runs both serial and parallel jobs. However, the Backfill

New Job Command Language Keywords

The following are new job command language keywords.

- network
- node
- node_usage
- tasks_per_node
- total_tasks

See “Job Command File Keywords” on page 38 for more information.

Administration File Changes

The following are new functions added to the administration file:

- New adapter stanza: this new stanza allows you to configure and use adapters. See “Step 5: Specify Adapter Stanzas” on page 93 for more information.
- New machine stanza keywords:
  - adapter_stanzas
  - pool_list
  - spacct_excluse_enable
  - machine_mode

  See “Step 1: Specify Machine Stanzas” on page 73 for more information.
- New user, group, and/or class stanza keywords:

Resource Manager Changes

LoadLeveler now incorporates functions which were previously part of the Parallel

Migration Considerations

Adapter keyword and that use the “==” operator are converted; all other **Adapter**

Changes in the GUI Resource File

The following new resources have been added to **Xloadl**, the GUI resource file:

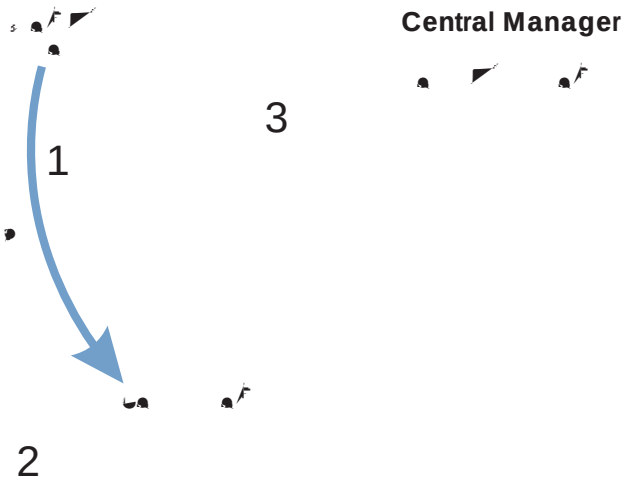
- New resources ending in _**label** allow you to specify the titles of the columns on the Jobs and Machines windows.
- Additional resources ending in _**len** allow you to add new fields to the Jobs and Machines windows and to specify the size of these fields.

Overview of LoadLeveler

Chapter 1. What is LoadLeveler?

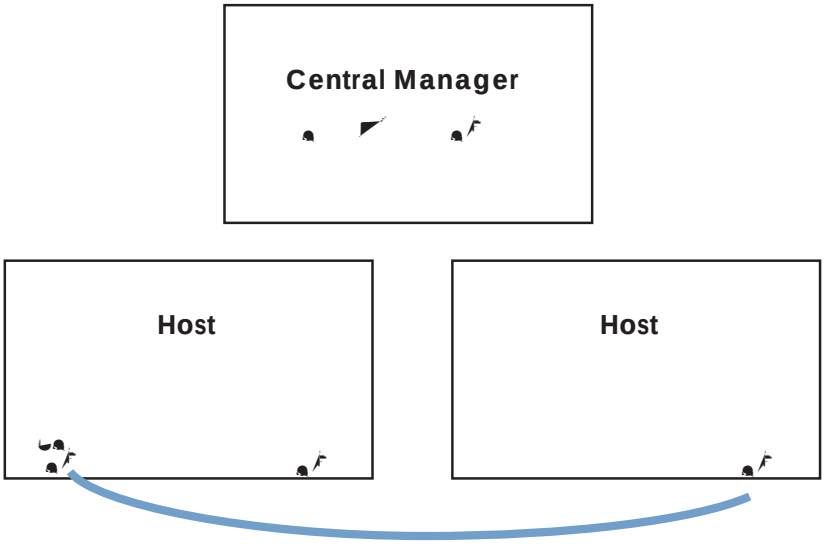
Machines and Workstations

Machine B


Central Manager
3
1
2

a *permit to run*. Once this is done, the job is considered Pending or Starting. (See “LoadLeveler Job States” on page 18 for more information.)

3. Prepare to run:



Chapter 2. LoadLeveler Daemons and Job States

This chapter presents a detailed explanation of LoadLeveler daemons and processes. Included here is a description of job states, which are controlled by certain daemons. See “LoadLeveler Job States” on page 18 for more information.

Daemons and Processes

This section presents a detailed explanation of LoadLeveler daemons and processes. For more information on configuration file keywords mentioned in this section, see “Configuring LoadLeveler” on page 94.

The master Daemon

The **master** daemon runs on every machine in the LoadLeveler cluster, except the submit-only machine. The real and effective user ID of this daemon must be root.

The master daemon determines whether to start any other daemons by checking the **START_DAEMONS** keyword in the global or local configuration file. If the keyword is set to **true**, the daemons are started. If the keyword is set to

- Re-run the **.llrc** file, reread the configuration files, stop or start daemons as appropriate for the new configuration files
-

- Records the following information in LoadLeveler variables and sends the information to the negotiator. These variables are described in “LoadLeveler Variables” on page 122.
  - State (of the startd daemon)
  - EnteredCurrentState
  - Memory
  - Disk
  - KeyboardIdle
  - Cpus
  - LoadAvg
  - Machine
  - Adapter
  - AvailableClasses
- Calculates the SUSPEND, RESUME, CONTINUE, and VACATE expressions.

- Query about jobs
- Remove a job
- Hold or release a job
- Favor or unfavor a user or a job.

-

Chapter 3. Submitting and Managing Jobs

This chapter tells you how to submit jobs to LoadLeveler. In general, the

- Statement components are separated by blanks. You can use blanks before or

LOADL_STEP_INITDIR	The initial working directory.
LOADL_STEP_NAME	The name of the job step.
LOADL_STEP_NICE	The UNIX *nice* value of the job step. This value is determined by the **nice** keyword in the class stanza. For more information, see “Step 3:

Querying the Status of a Job Running on an NQS Machine: If your job command file was routed to an NQS machine for processing, you can obtain its status by using either the GUI or the **llq**

Placing and Releasing a Hold on a Job

The **SYSPRIO** expression can contain class, group, and user priorities, as shown in the following example:

- **$(stepid)** refers to the job step identifier and increases after each **queue** command. Therefore, you only need to specify input, output, and error statements once to have LoadLeveler name these files correctly.

Assume that you created five input files and each input file has different initial conditions forifferen èà $ ã s f° lnents to the input files fthe to th rem ch r n¿i five tintt, each tin . Titb fs different input fil . eLoadLeveler ¸enerates th output file èà th reLoadLeveler job step th .t th reresuŸts èà

In Job Step 0:

- LoadLeveler creates the subdirectories **oslsslv_out**

Example 3: Using the Job Command File as the Executable

The name of the sample script shown in Figure 16 on page 38 is

data_limit

```
dependency = (step1 == □)
```

Example 2: In the following example, step1 will run with the executable called **myprogram1**. Step2 will run only if LoadLeveler removes step1 from the system. If step2 does run, the executable called **able call**[illegible]

$var	Specifies that the environment variable

image_size = *number*

files. Note that if you specify in a job command file both the **min_processors** keyword and the **node**

When you use the **node** keyword together with the **total_tasks** keyword, the *min* and *max* values you specify on the **node** keyword must be equal, or you must specify only one value. For example:

```
node = 6
total_tasks = 12
```

For information of specifying the number of tasks you want to run on a node, see “tasks_per_node” on page 55 and “total_tasks” on page 55.

node_usage

Specifies whether this job step shares nodes with other job steps.

The syntax is:

node_usage = shared | not_shared

Where:

shared Specifies that nodes can be shared with other tasks of other job steps. This is the default.

not_shared Specifies that nodes are not shared: no other job steps are scheduled on this node.

notification

Specifies when the user specified in the **notify_user** keyword is sent mail. The syntax is:

notify_user

Specifies the user to whom mail is sent based on the **notification** keyword. The default is the submitting user and the submitting machine.

The syntax is:

notify_user = *userID*

For example, if you are the job step owner but you want a co-worker whose name and user ID is **bob**, to receive mail regarding the job step, your notify keyword would be:

```
notify_user = bob
```

output

Specifies the name of the file to use as standard output (stdout) when your job step runs. If not specified, the file **/dev/null** is used.

The syntax is:

output = *filename*

For example:

```
output = out.$(jobid)
```

parallel_path

Specifies the path that should be used when starting a PVM 3.3 slave process. This is used for PVM 3.3 only and is translated into the **ep** keyword as defined in the PVM 3.3 **hosts** file.

For example:

```
parallel_path = /home/userid/cmds/pvm3/$PVM_ARCH:$PVM_ROOT/lib/$PVM_ARCH
```

The **parallel_path** statement above has two components, separated by a colon. The first component points to the location of the user's programs. The second component points to the location of the **pvmgs** routine – required if the job uses PVM 3.3 group support – assuming PVM 3.3 is installed “normally”. Note that your installation must install PVM 3.3 to include group support in order for you to use group support within LoadLeveler. $PVM_ARCH will be replaced by the architecture of the machine, as defined by PVM 3.3. This will specify the path to be searched for executables when the user's job issues a **pvm_spawn()** command.

$PVM_ARCH, and $PVM_ROOT are PVM environment variables. For more information, see the appropriate PVM 3.3 documentation.

preferences

Specifies the characteristics that you prefer be available on the machine that executes the job steps. LoadLeveler attempts to run the job steps on machines that meet your preferences. If such a machine is not available, LoadLeveler will then assign machines which meet only your requirements.

The values you can specify in a **preferences** statement are the same values you can specify in a **requirements** statement, with the exception of the **Adapter** requirement. See “requirements” on page 51 for more information.

The syntax is:

preferences = *Boolean_expression*

Note that LoadLeveler converts the above network types to the **network** statement. For more information, see “Migrating Your Existing Adapter Requirements Statements” on page xix.

Arch Specifies the machine architecture on which you want your job step to run. It describes the particular kind of UNIX platform for which your executable has been compiled. The default is the architecture of the submitting machine.

Disk Specifies the amount of disk space in kilobytes you believe is required in the LoadLeveler **execute** directory to run the job step.

Feature Specifies the name of a feature defined on a machine where you want your job step to run. Be sure to specify a feature in the same way in which the feature is specified in the machine stanza of the administration file. To find out what features are available, use the **llstatus** command.

```
# @ step_name    = step1
# @ executable   = c1
# @ output       = $(executable).$(jobid).$(step_name).out
# @ queue
# @ step_name    = step2
# @ dependency   = (step1 == ▫)
# @ requirements = (Machine == machine.step1)
# @ executable   = c2
# @ output       = $(executable).$(jobid).$(step_name).out
# @ queue
```

Example 4: To specify a requirement for an SP switch adapter in IP mode, enter:

```
requirements = (Adapter == "hps_ip")
```

shell

Specifies the name of the shell to use for the job step. If not specified, the shell used in the owner's password file entry is used. If none is specified, the /bin/sh is used.

The syntax is:

For example, to run two tasks on each of 12 available nodes for a total of 24 tasks,

The following variables are available to you:

Chapter 4. Submitting and Managing Parallel Jobs

This chapter tells you how to submit and manage parallel jobs. For information on setting up and planning for parallel jobs, see Chapter 6, “Administration Tasks for Parallel Jobs” on page 135.

Supported Parallel Environments

LoadLeveler allows you to schedule parallel batch jobs that have been written using the following:

- IBM Parallel Environment Library* (POE/MPI/LAPI) 2.4.0
- Parallel Virtual Machine (PVM) 3.3 (RS6K architecture)
- Parallel Virtual Machine (PVM) 3.3.11+ (SP2MPI architecture)

Note that for parallel batch jobs, LoadLeveler no longer interacts with the PSSP

PVM 3.3.11+ (SP2MPI architecture)

additional task will be spawned on the last processor in the LOADL_PROCESSOR_LIST. For more information on this environment variable set by LoadLeveler see “Obtaining Allocated Host Names” on page 64.

- You must use the PVM daemon and starter path dictated by the LoadLeveler administrator. The **parallel_path**

- You must *

Administering LoadLeveler

Chapter 5. Administering and Configuring LoadLeveler

This chapter tells you how to administer and configure LoadLeveler. In general, the information in this chapter applies to both serial and parallel jobs. For more specific information on parallel jobs, see Chapter 6, “Administration Tasks for Parallel Jobs” on page 135.

Overview

After installing LoadLeveler, you need to customize it by modifying both the *administration* file and the *configuration* file. The administration file optionally lists and defines the machines in the LoadLeveler cluster and the characteristics of

```
machine_a: type = machine
      central_manager = true    # defines this machine as the central manager
```

central_manager = true**| false |** alt

where **true** designates this machine as the LoadLeveler central manager host,

name for machine names specified in LoadLeveler commands issued from the machine described in this stanza.

If DNS is specified alone, LoadLeveler will always append the DNS domain name to machine names specified in LoadLeveler commands. If NIS or LOCAL is specified, LoadLeveler will never append a DNS domain name to machine names specified in LoadLeveler commands. If DNS is specified with either NIS or LOCAL, LoadLeveler will always look up the name in the administration file to determine whether to append a DNS domain name. If the name is specified with a trailing period, it doesn't append the domain name.

pvm_root = *pathname*

Where *pathname* specifies the location o f the L n the et pphname

account =*list*

where *list* is a blank-delimited list of account numbers that identifies the

total_tasks = *number*

where *number* specifies the maximum number of tasks this user can request for a parallel job in a job command file using the **total_tasks**

excluded groups. Only one of these may be used for any class. The default is that no groups are excluded.

exclude_users = *list*

where *list* is a one è¢ èà ¸ è] fort è] f thers =

This value ranges from -20 to 20. Values out of this range are placed at the top (or bottom) of the range. For example, if your current *nice*

Exceeding Per Process Limits: For per process limits, what happens when your job reaches and exceeds either the soft limit or the hard limit depends on the

You can use the following character strings with all limit keywords:

rlim_infinity	Represents the largest positive number.
unlimited	Has same effect as **rlim_infinity**.
copy	Uses the limit currently active when the job is submitted.

See Table 8 for more information on specifying limits.

```
cpu_limit = 12:56:21       # hardlimit = 12 hours 56 minutes 21 seconds
cpu_limit = 56:□□,5□:□□    # hardlimit = 56 minutes □ seconds
# softlimit = 5□ minutes □ seconds
cpu_limit = 1:□3           # hardlimit = 1 minute 3 seconds
cpu_limit = unlimited      # hardlimit = 2,147,483,647 seconds
# (X'7FFFFFFF')
cpu_limit = rlim_infinity  # hardlimit = 2,147,483,647 seconds
# (X'7FFFFFFF')
cpu_limit = copy           # current CPU hardlimit value on the
# submitting machine.
```

restarted, the **wall_clock_limit** is no ¸ è_ ,] n rt t e)è¢ è_ , ik

maxqueued =

The Configuration Files

By taking a look at the configuration files that come with LoadLeveler, you will find that there are many parameters that you can set. In most cases, you will only have to modify a few of nd

The Backfill scheduler supports:

- The scheduling of multiple tasks per node.

Step 5: Prioritize the Queue Maintained by the Negotiator

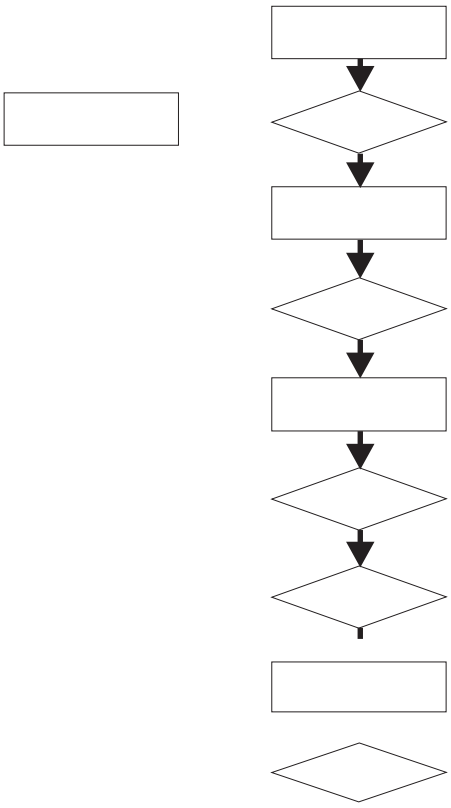
Each job submitted to LoadLeveler is assigned a system priority number, based on the evaluation of the **SYSPRIO** keyword expression in the configuration file of the central manager. The LoadLeveler system priority number is assigned when the central manager adds the new job to the queue of jobs eligible for dispatch. Once assigned, the system priority number for a job is never changed (unless jobs for a user swap their **SYSPRIO**, or **NEGOTIATOR_RECALCULATE_SYSPRIO_INTERVAL** is not zero). Jobs assigned higher **SYSPRIO** numbers are considered for dispatch before jobs with lower numbers. See “How Does a Job's Priority Affect Dispatching Order?” on page 30 for more information on job priorities.

You can use the follow $ Eac« $ Eac´jo ng fok n us ighk)è¢ ã´ èÃ èà SYSPRIO è

- You can use the

Mefi

negotiator assigns a job. For more information, see the



A_DETAIL

Step 11: Record and Control Log Files

The LoadLeveler daemons and processes keep log files according to the

If LoadLeveler does not find the value in the configuration file, it looks in the **/etc/services** file. If the value is not found in this file, the default is used.

The configuration file keywords ass l r t d it– ”ort zudsers arn the oo lo in :)è¢

Avoid Using Certain System Services in Checkpointed Jobs: In order to

```
crxlc myprog myprog.c
```

also negotiates with machines whenever job states or machine states change. The default is 30 seconds.

NEGOTIATOR_LOADAVG_INCREMENT

where *number* specifies the value the negotiator adds to the startd machine's load average whenever a job in the Pending state is queued on that machine. This value is used to compensate for the increased load caused by starting another job. The default value is .5.

wmpne. r
c è c j vØ lud è s e. :, t Œ Lú¢ è. e è
tú¢ ud è s e. e Ú l s t a N l s « l u è t $ o v

OBITUARY_LOG_LENGTH = *number*

where *number* specifies the number of lines from the end of the file that are appended to the mail message. The master daemon mails this log to the

A = xxx‰j 2@”7¥”“‰j 04@•47’j 25j 04@£ˆ•@™•47¥”“ –j 06@@‰¢

Configuration File Keyword	Brief Description	For Details
JOB_LIMIT_POLICY	The amount of time in seconds that LoadLeveler checks to see if **job_cpu_limit** has been exceeded.	Chapter 7, “Gathering Job Accounting Data” on page 141
JOB_PROLOG	Pathname of the prolog program.	“Writing Prolog and Epilog Programs” on page 280
JOB_USER_EPILOG	Pathname of the user epilog program.	“Writing Prolog and Epilog Programs” on page 280
JOB_USER_PROLOG	Pathname of the user prolog program.	“Writing Prolog and Epilog Programs” on page 280
KBDD	KBDD expression. Location of kbdd executable (Loadl_kbdd).	“LoadLeveler Daemons” on page 6
KILL	Kill expression. Determines if vacated jobs should be killed.	“Step 7: Manage a Job's Status Using Control Expressions” on page 105
LIB	The directory where LoadLeveler libraries are kept.	“Step 10: Specify Where Files and Directories are Located” on page 109
LOADL_ADMIN	List of LoadLeveler administrators.	“Step 1: Define LoadLeveler Administrators” on page 96Peci $ P ll % ´ gra 9 ´ ¸ el$

Configuration File Keyword	Brief Description	For Details
MAX_CKPT_INTERVAL	The maximum number of seconds	

Keyword	Brief Description
CPU_Busy	Defines the variable **CPU_Busy** and reassigns to it at each evaluation the Boolean value True or False, depending on whether the Berkeley one-minute load average is equal to or greater than the saturation level of 1.5.
CPU_Idle	

Chapter 6. Administration Tasks for Parallel Jobs

This chapter describes administration tasks that apply to parallel jobs. For more

```
k1▫n▫9.ppd.pok.ibm.com: type = adapter
 adapter_name = en▫
 network_type = ethernet
 interface_address = 9.114.51.73
 interface_name = k1▫n▫9.ppd.pok.ibm.com

k1▫sn▫9.ppd.pok.ibm.com: type = adapter
 adapter_name = css▫
 network_type = switch
 interface_address = 9.114.51.137
 interface_name = k1▫sn▫9.ppd.pok.ibm.com
 switch_node_number = 8

k1▫n▫9.ppd.pok.ibm.com: type=machine
 adapter_stanzas = k1▫n▫9.ppd.pok.ibm.com k1▫sn▫9.ppd.pok.ibm.com
 spacct_exclusive_enable = true
```

- A job can request a maximum of 60 nodes and 120 total tasks
- Jobs will have a relatively low run priority

Chapter 7. Gathering Job Accounting Data

Your organization may have a policy of charging users or groups of users for the amount of resources that their jobs consume. You can do this using LoadLeveler's accounting feature. Using this feature, you can produce accounting reports that contain job resource information for completed serial and parallel jobs. You can

@
@

@Ÿ ä¢…™ ••”…

Ÿ ä¢…™}¢@“–‡‰•@‡™–¤—@••”…

Ÿ Áƒƒ–¤•£@•¤”,…™@¢—…ƒ‰†‰…„@–•@£ˆ…@Ñ–,

Ÿ Â“••’@¢…—•™•£…„@“‰¢£@–†@•ƒƒ–¤•£@•¤”,…™¢@–,£•‰•…„@†™–”@£ˆ…@¤¢…™}¢@
•„”‰•‰¢£™•£‰–•@†‰“…K

ãˆ…@•ƒƒ–¤•£@¥•“‰„•£‰–•@™–¤£‰•…@”¤¢£@…§‰£@¦‰£ˆ@•@™…£¤™•@ƒ–„…@–†@©
¢¤ƒƒ……„¢K@É†@‰£@†•‰“¢k@£ˆ…@™…£¤™•@ƒ–„…@‰¢@•@•–•`©…™–@•¤”,…™K

Ã–““…ƒ£‰•‡@£ˆ…@Áƒƒ–¤•£‰•‡@É•†–™”•£‰–•@••„@â£–™‰•‡@‰£@‰•£–@Æ

Ó–•„Ó…¥…“…™@¢£–™…¢@£ˆ…@•ƒƒ–¤•£‰•‡@‰•†–™”•£‰–•@£ˆ•£@‰‰@@ƒ™“¨…ƒ£¢@‰•(
‰•@£ˆ…@¢——“@„‰™…ƒ£–™¨@–†@£ˆ…@”•ƒˆ‰•…@£ˆ•£@‰•‰£‰•“““¨@¢ƒˆ…„¤“…„@£ˆ%
”•ƒˆ‰•…K@Ä•£•@–•@—•™•““…“@‘–‚¢@•™…@•“¢–@¢£–@†‰„@‰K@£ˆ…@

Ù…¢–¤™ƒ…@‰•†–™”•£‰–•@ƒ–““…ƒ£…„@–•@£ˆ…@Ó–•„Ó…¥…“…™@‘–‚@‰¢@ƒ–•¢£™•%

Task 3: Report Job Information on all the Jobs in the History File

You can accomplish this step using either the

Task 5: Specifying Machines and Their Weights

To specify weights to associate with machines, specify the following keyword in a machine's machine stanza in the administration file:

speed = *number*
where *number* defines the weight associated with a particular machine. The higher numbers correspond with a greater weight. The default weight is 1.0.

Also, if you have in your cluster machines of differing speeds and you want LoadLeveler accounting information to be normalized for these differences, specify **cpu_speed_scale=true** in each machine's respective machine stanza.

For example, suppose you have a cluster of two machines, called A and B, where Machine B is three times as fast as Machine A. Machine A has **speed=1.0**, and Machine B has **speed=3.0**. Suppose a job runs for 12 CPU seconds on Machine A.

notify_user	converted to **-mu** option
output	converted to **-o** option
preferences	used only for LoadLeveler scheduling
queue	

Chapter 9. LoadLeveler Commands

LoadLeveler provides two types of commands: those that are available to all users of LoadLeveler, and those that are reserved for LoadLeveler administrators. (Administrators are identified by the **LOADL_ADMIN** keyword in the configuration file.)

The administrator commands can operate on the entire LoadLeveler job queue and all machines configured. The user commands mainly affect those jobs submitted by that user. Some commands, such as **llhold**, include options that can only be performed by an administrator.

Summary of LoadLeveler Commands

The following table summarizes the LoadLeveler commands:

Command	Description	Who Can Issue?	For More Information
llacctmrg	Collects all individual machine history files together into a single file.	Administrators	See page 156
llcancel	Cancels a submitted job.	Users and Administrators	See page 158
llclass	Returns information about LoadLeveler classes.	Users and Administrators	See page 16

llacctmrg - Collect machine history files

Purpose

Collects individual machine history files together into a single file specified as a parameter.

Syntax

llacctmrg [

Examples

The following example collects data from machines named `mars` and `pluto`.

```
llacctmrg -h mars pluto
```

The following example collects data from the machine named `mars` and places the data in an existing directory called **merge**.

```
llacctmrg -h mars -d merge
```

Results

The following shows a sample system response from the **llacctmrg -h mars -d merge** command.

```
llacctmrg: History transferred successfully from mars (1□□8□ bytes)
```

llclass - Query Class Information

Purpose

Returns information about classes.

Syntax

llclass [**-?**] [**-H**] [**-v**] [**-l**] [*classlist*]

Flags

-? Provides a short usage message.

-H Provides entended help information.

-v Outputs the name of the command, release number, service level, service level date, and operating system used to build the command.

-l Specifies that a long listing be generated for each class for which status is requested. If **-l** is *not* specified, then the standard listing is generated.

classlist Is a blank-delimited list of classes for which you are requesting status. If no *classlist* is specified, all classes are queried.

If you have more than a few classes configured for LoadLeveler, consider redirecting the output to a file when you use the **-l** flag.

Examples

g e x a m p l e s f i l e w t u s . l ´ ´ ¸ ´

llclass

llctl - Control LoadLeveler Daemons

Purpose

Controls LoadLeveler daemons on all members of the LoadLeveler cluster.

Syntax

llctl [**-?**] [**-H**] [**-v**] [**-q**] [**-g** |

If you need to return the machine to the cluster later, you must clear all files from the spool and execute directory of the machine which was deleted.

This example resumes the classes

llextSDR - Extract adapter information from the SDR

Purpose

Extracts adapter information from the system data repository (SDR) and creates

```
k1▫sn▫9.ppd.pok.ibm.com: type = machine
 adapter_stanzas = k1▫sn▫9.ppd.pok.ibm.com k1▫n▫9.ppd.pok.ibm.com
 spacct_excluse_enable = true

k1▫sn▫9.ppd.pok.ibm.com: type = adapter
 adapter_name = css▫
 network_type = switch
 interface_address = 9.114.51.137
 interface_name = k1▫sn▫9.ppd.pok.ibm.com
 switch_node_number = 8

k1▫n▫9.ppd.pok.ibm.com: type = adapter
 adapter_name = en▫
 network_type = ethernet
 interface_address = 9.114.51,73
 interface_name = k1▫n▫9.ppd.pok.ibm.com
```

llfavorjob - Reorder System Queue by Job

Purpose

llhold

joblist

llinit - Initialize Machines in the LoadLeveler Cluster

llprio

llq - Query Job Status

Purpose

Returns information about jobs that have been dispatched.

Syntax

llq

%nh Number of hosts allocated to the job
%o Job owner
%p User priority
%sn Step name
%st Status

-r *category_list* Is a blank-delimited list of formats (categories) you want to query. Each category you specify must be preceded by a percent sign. The *category_list* cannot contain duplicate entries. This flag

llq

Customized, Unformatted Standard Listing: A customized and unformatted (raw) standard listing is generated when you specify **llq** with the **-r** flag. Output fields are separated by an exclamation point (!). The following is sample output from this command:

```
llq -r %id %c %dq %dd %gl %h
```

```
ll6.pok.ibm.com.2.▫!No_Class!▫4/▫8 ▫9:19!▫4/▫8 ▫9:21!No_Group!ll6.pok.ibm.com
ll6.pok.ibm.com.1.▫!No_Class!▫4/▫8 ▫9:19!▫4/▫8 ▫9:21!No_Group!ll6.pok.ibm.com
ll6.pok.ibm.com.3.▫!No_Class!▫4/▫8 ▫9:19!▫4/▫8 ▫9:21!No_Group!ll5.pok.ibm.com
```

The Long Listing: The long listing is generated when you specify the **-l** option with the **llq** command. This section contains sample output for two **llq** commands: one querying a serial job and one querying a parallel job. Following the sample output is an explanation of all possible fields displayed by the **llq** command.

The following is sample output for the **llq -l** command for the serial job "ll6.pok.ibm.com.2."

llq

Master Task
Executable : /u/richc/sp/poe/poe.musppa
Exec Args : /u/richc/sp/mpi/fvt_mpi -vw┘├┐cccØgbwehE -├┐ ┬├▫lee┘▫hE ┐├e┘ ┐├sk)è¢ ┐├h

Completion Date date and time job completed or exited.

Completion Code the status returned by the wait3 UNIX system call.

User Priority priority of the job, as specified by the user in the job command, or changed by the **llprio** command.

user_sysprio user system priority of the job, where the value is defined in the administration file.

class_sysprio class priority of the job, where the value is defined in the administration files.

group_sysprio group priority of the job, where the value is defined in the administration files.

System Priority overall system priority of the job, where the value is defined by the SYSPRIO expression in the configuration file.

q_sysprio adjusted system priority of the job (See “How Does a Job's Priority Affect Dispatching Order?” on page 30.)

Notifications notification status for the job, where:

always
indicates notification is sent through the mail for all four

Init Working Directory

directory from which the job is r ´ èÃ _Ä k job j b tory

InQ

StartdAvail flag indicating if machine is running a startd daemon (0=no, 1=yes).

State state of the startd daemon, which can be:

Busy
Down
Drain
Flush
Idle
None
Reserved
Running
Suspend

For a detailed explanation of these states, see “The startd Daemon” on page 15.

OpSys operating system on this machine.

Arch hardware architecture of machine as listed in configuration file.

Machine Fully qualified name of this machine's host.

Total Jobs	number of total job steps submitted to this schedd machine.
ScheddState	state of the schedd on this schedd machine.
time stamp	date and time the central manager last received a status update from this schedd machine.

llsummary - Return Job Resource Information for Accounting

Purpose

Returns job resource information on completed jobs for accounting purposes.

Syntax

llsummary [**-?**] [**-H**] [**-v**] [**-x**] [**-l**] [**-s**

-r *report*

llsummary

The following is an example of the standard listing:

```
       Name   Jobs   Steps        Job Cpu    Starter Cpu     Leverage
    krystal     15      36    ¤+¤¤â┘─┘─├ ┘t¤¸ ¤¤)├ ¤─┘¤´¤¤¸¤┘¤´¤¤┬ ─┐─o┐â┘ ├.¤´¤¤├ae
```

llsummary

Getting Help Using the Graphical User Interface

You can get help when using the GUI by pressing the Help key. This key is function key 1 (F1) on most keyboards. To receive help on specific parts of the LoadLeveler GUI, place the cursor over the area or field on which you want help and press F1. A help screen appears describing that area. You can also get help by using the Help pulldown menu and the Help push buttons available in pop-up windows.

Before you invoke the GUI, make sure your PATH statement includes the directory

Selections from a menu bar are indicated with an →. For example, if a menu bar
inclv id wn . wnclv id wn

SELECT a Restart option

Field	Input
CPU Limit	Maximum amount of CPU time that the submitted job can use. Express the amount as:

Step 2: Edit the Job Command File

There are several ways that you can edit the job command file that you just built:

If you already submitted your job, go to “Step 4: Display, Refresh and Obtain Job

cycles. You should specify a refresh interval of 120 seconds or more for normal use.

SELECT OK

□

SELECT Actions → Cancel

 A warning dialog box appears prompting you to confirm your cancellation request. Once you confirm your request, LoadLeveler cancels the job and the job information disappears from the Jobs window.

Go to the next step.

Step 10: Display and Refresh Machine Status

Each sorting option contains a cascading window which allows you to select this

To	Select Select →
View all of the machines	**All**

Select **Select → Show Selection** to show the selection parameters.

Go to the next step.

in the **llhold**, **llfavoruser**, and **llfavorjob** commands. The **Admin** pulldown menu on the Machines window corresponds to the command options available in the **llctl** command.

The main window of the GUI, as shown in Figure 32 on page 212, has three sub-windows: one for job status with pull-down menus for job-related commands, one for machine status with pull-down menus for machine-related commands, and one for messages and logs. There are a variety of facilities available that allow you to sort and select the items displayed.

Job Related Administrative Actions

You access the administrative commands that act on jobs through the **Admin** pulldown menu in the Jobs window of the GUI.

´ ´ è_ v bs v ad ¸ h ´ ´ Jo

reconfig forces all daemons to reread the configuration files.

Select the machine on which you want to operate. To reconfigure this **xloadl** session, choose **reconfig** but do not select a machine.

Select **Admin** from the Machines window.

Select **reconfig**.

recycle stops all LoadLeveler daemons and restarts them.

Select the machine on which you want to operate.

Select **Admin** from the Machines window.

Select **recycle**.

drain allows no more LoadLeveler jobs to begin running on this machine but it does allow running jobs to complete.

Select the machine on which you want to operate.

Select **Admin** from the Machines window.

Select **drain**.

A cascading menu allows you to select either **daemons**, **schedd**, **startd**, or **startd by class**. If you select **daemons**, both machines will be drained. If you select **schedd**, only the schedd on the selected machine will be drained. If you select **startd**, only the startd on the selected machine will be drained. If you select **startd by class**, a window appears which allows you to select classes to be started.

flush terminates running jobs on this host and sends them back to the system queue to await redispatch. No new jobs are redispatched to this machine until **resume** is issued. Forces a checkpoint if jobs are enabled for checkpointing.

Select the machine on which you want to operate.

Select **Admin** from the Machines window.

Select **flush**.

suspend suspends all jobs on this host.

Select the machine on which you want to operate.

Select **Admin** from the Machines window.

Select

you select **daemons**, both machines will be

The LoadLeveler Application Programming Interfaces

Chapter 11. LoadLeveler APIs

Return Values

GetHistory returns a zero when successful.

Error Values

GetHistory returns -1 to indicate that the version is not supported or that an error occurred opening the history file.

Examples

Makefiles and examples which use this API are located in the **samples/llphist**

The Submit API

This API allows you to submit jobs to LoadLeveler. The submit API consists of the **llsubmit**

Return Values

0 The job was submitted.

Error Values

-1 The job was not submitted. Error messages are written to stderr.

llfree_job_info Subroutine

Purpose

llfree_job_info frees space for the array and the job step information used by **llsubmit**.

Syntax

void llfree_job_info(LL_job ▫*job_info*

state

Is the current state of the job step. Possible values for the state are:

Library

LoadLeveler API library **libllapi.a**

Syntax

returned by the **llq -f** flag. For more information, see “llq - Query Job Status” on page 181.

- If you specify MACHINES, *data_filter* can be ALL_DATA (the default), which returns the entire object, or STATUS_LINE, which returns the same information returned by the **llstatus -f** flag. For more information, see “llstatus - Query Machine Status” on page 193.

Description

query_element,

Related Information

Subroutines: **ll_get_data**, **ll_query**, **ll_reset_request**, **ll_get_objs**, **ll_free_objs**, **ll_next_obj**, **ll_deallocate**.

ll_reset_request Subroutine

Purpose

The **ll_reset_request** subroutine resets the request data to NULL for the

Syntax

```
#include "llapi.h"

LL_element ▫ ll_get_objs(LL_element ▫query_element,LL_Daemon query_daemon,
char ▫hostname,int ▫number_of_objs,int ▫error_code);
```

Parameters

query_element

Is a pointer to the **LL_element** returned by the **ll_query** function.

query_daemon

Error Values

This subroutine a NULL to indicate failure. The *error_code* parameter is set to one of the following:

-1

The Task object represents one or more copies of the same executable. The Task object can be queried for the executable, the executable arguments, and the number of instances of the executable.

Table 11 describes the specifications and elements available when you use the **ll_get_data** subroutine. Each specification name describes the object you need to specify and the attribute returned. For example, the specification LL_JobGetFirstStep includes the object you need to specify (LL_Job) and the value returned (GetFirstStep).

This table is sorted alphabetically by object; within each object the specifications are also sorted alphabetically. tn os e e _get_rs ã c _ cutabl èà è èà ces å ¢

Table 11 (Page 5 of 5). Specifications for ll_get_data Subroutine

Specification	Object	Resulting Data Type	Description
LL_StepRssLimitSoft	Step	int*	A pointer to an integer indicating the RSS soft limit set by the user in the **rss_limit** keyword.
LL_StepShell	Step	char*	A pointer to a character string containing the shell name used by the executable.
LL_StepStackLimitHard	Step	int*	A pointer to an integer indicating the stack hard limit set by the user in the **stack_limit** keyword.
LL_StepStackLimitSoft	Step	int*	A pointer to an integer indicating the stack soft limit set by the user in the **stack_limit** keyword.
LL_StepStartCount	Step	int*	A pointer to an integer indicating the number of times the step has been started.
LL_StepStartDate	Step	time_t*	A pointer to a time_t structure indicating the value the user specified in the **startdate**

Library
LoadLeveler API library **libllapi.a**

Syntax
```
#include "llapi.h"
```

ll_next_obj Subroutine

Purpose

The **ll_next_obj** subroutine returns the next object in the *query_element* list you specify.

Parameters

query_element

Is a pointer to the **LL_element** returned by the **ll_query** function.

Examples of Using the Data Access API

Example 1: The following example obtains a list of current job objects from the

```
main(int argc,char ▫argv[])
{
```

Example 3: The following example queries information about the hosts k10n11 and k10n06:

```
#include "llapi.h"

main(int argc,char ▫argv[])
{
  LL_element ▫queryObject=NULL, ▫machine=NULL;
  int rc, num, err;
```

– The Parallel Master receives a VACATE or FLUSH request.

The SIGUSR1 is also sent to all parallel tasks.

A SIGKILL is issued to any process which does not exit within two minutes of receiving a termination signal.

ll_get_hostlist Subroutine

Purpose

This subroutine obtains a list of machines from the Master Starter machine so that the Parallel Master can start the Parallel t

Description

This function must be invoked for all the machines returned from the **ll_get_hostlist**

The syntax to invoke the Parallel M t‰ Ä r is:

para_api

The syntax to invoke the Parallel Slave is:

para_api -s

The Parallel M t‰ Ä r do s the following:

- t quir s the hos Ä « ‰ Ä thro**Ä** h_the**t_ os**t **Ä**and p**‰ä**s ou Ä the r tu ned fields.
- ‰ Ä rts a Parallel Slave task by x cu Äing the com t d sp cified in the **‰ Ä rtCm**ri t le on all hos Ä r tu ned in the hos Ä « ‰ Ä.
- t quir s the sock t connec Ä d to the Parallel Slave's s Ä ndard I/O (s Ä io).
- ri Ä s a com t d ov r the sock t to v rify s Ä in.
- R t s acknow Ä d m nts ov r the sock t to v rify s Ä rr and s Ä ou Ä.
- Prints ou Ä hos Ä t s and acknow Ä d m nts.

xb le ou Äpu Ä follows:

```
nu┘▫┘ _nod┘▫s=2

nt┘ ┘▫=hos├ Ä─▫├.kgn.ibm.com t┐ drss=9.115.8.162 swiÄ┐─h_nu┘▫┘ br=-1

nt┘ ┘▫=hos├ Ä─▫┘.kgn.ibm.com t┐ drss=9.115.8.164 swiÄ┐─h_nu┘▫┘ br=-1

Connec├ Ä┐d to hos├ Ä─▫├.kgn.ibm.com t Ä sock ─▫─
RÄ┐ived acko "8─▫┐─▫┐─▫┐├  and acke "1▫─▫┐─▫┐─▫┐├  from hos├ Ä ─▫┐

Connec├ Ä┐d to hos├ Ä─▫┘.kgn.ibm.com t Ä sock ─▫
RÄ┐ived acko "8─▫┐─▫┐─▫├├  and acke "1▫─▫┐─▫┐─▫├├  from hos├ Ä ─▫├
```

Description

You must set **SCHEDULER_API = YES** in the global configuration file to use this subroutine.

Only jobs steps currently in the Idle state are started.

Only processes having the LoadLeveler administrator user ID can invoke this subroutine.

An external scheduler uses this subroutine in conjunction with the **ll_get_nodes** and **ll_get_jobs** subroutines of the query API. The query API returns information about which machines are avialable for scheduling and which jobs are currently in the job queue waiting to be scheduled.

Return Values

This subroutines return a value of zero to indicate the start job request was accepted by the negotiator. However, a return code of zero does not necessarily imply the job started. You can use the **llq** command to verify the job started.

Library

LoadLeveler API library **libllapi.a**

Syntax

```
#include "llapi.h"

int ll_get_jobs(LL_get_jobs_info □);
```

program2 receives the following as standard input:

- The length of the handle to follow
- The same handle written by *program1*

program2 writes the following to standard output:

- The length of the login context to follow
-

LOADL_STEP_ID	The step identifier, generated by LoadLeveler.
LOADL_JOB_CPU_LIMIT	The number of CPU seconds the job is limited to.
LOADL_WALL_LIMIT	The number of wall clock seconds the job is limited to.

LoadLeveler writes the following current AFS credentials, e ´ tep)è¢)è¢ job hs)è

Where *prolog_program* is the name of the prolog program as defined in the JOB_PROLOG keyword.

No arguments are passed to the program but several environment variables are set. These environment variables are described in “Submitting a Job Command File” on page 26.

The real and effective user ID of the prolog process is the LoadLeveler user ID. If the prolog program requires root authority, the administrator must write a secure C

```
#!/bin/csh
#
```

```
#!/bin/ksh
#
# Set up environment
set -a
. /etc/environment
. ~/.profile
export PATH="$PATH:/loctools/lladmin/bin"
export LOG="/tmp/$LOADL_STEP_OWNER.$LOADL_JOB_ID.epilog"
#
if [ [ -z $LOADL_PROLOG_EXIT_CODE ] ]
then
echo "Prolog did not run" >> $LOG
else
echo "Prolog exit code = $LOADL_PROLOG_EXIT_CODE" >> $LOG
fi
#
if [ [ -z $LOADL_USER_PROLOG_EXIT_CODE ] ]
  then
   echo "User environment prolog did not run" >> $LOG
  else
   echo "User environment exit code = $LOADL_USER_PROLOG_EXIT_CODE" >> $LOG
fi
#
if [ [ -z $LOADL_JOB_STEP_EXIT_CODE ] ]
  then
   echo "Job step did not run" >> $LOG
  else
   echo "Job step exit code = $LOADL_JOB_STEP_EXIT_CODE" >> $LOG
fi
#
#
# Do clean up based upon job step class
#
case $LOADL_STEP_CLASS in
  een
```

```
#!/bin/csh
#
# Set up environment
source /u/loadl/.login
#
setenv PATH  "${PATH}:/loctools/lladmin/bin"
setenv LOG "/tmp/${LOADL_STEP_OWNER}.${LOADL_JOB_ID}.prolog"
#
if ( ${?LOADL_PROLOG_EXIT_CODE} ) then
echo "Prolog exit code = $LOADL_PROLOG_EXIT_CODE" >> $LOG
else
echo "Prolog did not run" >> $LOG
endif
#
if ( ${?LOADL_USER_PROLOG_EXIT_CODE} ) then
    echo "User environment exit code = $LOADL_USER_PROLOG_EXIT_CODE" >> $LOG
  else
    echo "User environment prolog did not run" >> $LOG
endif
#
if ( ${?LOADL_JOB_STEP_EXIT_CODE} ) then
    echo "Job step exit code = $LOADL_JOB_STEP_EXIT_CODE" >> $LOG
  else
    echo "Job step did not run" >> $LOG
endif
#
# Do clean up based upon job step class
#
switch ($LOADL_STEP_CLASS)


#
```

Appendixes

Appendix A. Troubleshooting

Troubleshooting LoadLeveler

This chapter is divided into the following sections:

-

using LoadLeveler and LoadLeveler later attempts to start a job for that user on the same machine, LoadLeveler may not be able to start PVM for the job. This will cause the LoadLeveler job to be cancelled.

If a PVM job submitted through LoadLeveler is rejected, it is probably because PVM was not correctly terminated the last time it ran on the rejecting machine. LoadLeveler attempts to handle this by making sure that it cleans up PVM jobs when they complete, but remember that you may need to clean up after the job yourself. If a machine refuses to start a PVM job, check the following:

- See if there is a process with the name **pvmd** running on the machine in question under the id of the user whose job will not start. Stop the process by issuing:

  ps -ef | grep pvmd
  kill -TERM *pid*

  Do not use either of the following variations to stop the daemon because this will prevent **pvmd** from cleaning up and jobs will still not start:

  kill -9 *pid*
  kill -KILL *pid*

- If there is no **pvmd** process running, see if there is a file bt° n o

Why Doesn't LoadLeveler Execute my .profile or .login Script?: When you submit a batch job to LoadLeveler, the operating system will execute your **.profile** script before executing the batch job if your login shell is the Korn shell. On the other hand, if your login shell is the Bourne shell, on most operating systems

controls the number of jobs in the Idle state, since Pending and Starting are usually temporary states.

```
Off_Shift = ((tm_wday == 6) || (tm_wday == 0) || (tm_hour >=18) \
|| (tm_hour < 8))
```

Appendix B. Customer Case Studies

This chapter gives you an overview, including configuration information, of some LoadLeveler customers. These profiles are meant to highlight how customers in different industries use LoadLeveler.

Note that all of these configurations apply to Version 1 Release 3 of the default

```
#
# Specify pathnames
#
RELEASEDIR = /usr/lpp/LoadL/nfs
LOCAL_CONFIG = $(tilde)/local/configs/LoadL_config.$(host)
```

```
#
# Customized mail delivery program.
#
# MAIL =

#
# Customized submit (job command file) filter program.
#
```

```
#
# Negotiator Macros
#
NEGOTIATOR  = $(BIN)/LoadL_negotiator
```

```
#       NQS Directory
#
#
# For users of NQS resources:
# Specify the directory containing qsub, qstat, qdel
#
# NQS_DIR   = /usr/bin

#
# Specify Custom metric keywords
#
# CUSTOM_METRIC  =
# CUSTOM_METRIC_COMMAND = $(ETC)/sw_chip_number
#
# Machine control expressions and macros
#

OpSys :  $(OPSYS)
Arch  :  $(ARCH)
Machine :  $(HOST).$(DOMAIN)

#
# Expressions used to control starting and stopping of foreign jobs
#
MINUTE  = 6▫
HOUR  = (6▫ ▫ $(MINUTE)─╢)
#
R$(├┼#$(H   = (g ▫ :  ) STOQ▫Tè┌$ # C:┘w_▫{ $(├┼ è]è┌┘┘┐├) &&$)├ds è┐┘g ▫ :# è]è:├$(├┼



#├R¢è┌$ # Arch $─┐─)è¢┐  R¢è┌$┐▫s aD = ci┐)
```

#

Customer 2: Circuit Simulation

This customer performs CPU-intensive work in the area of circuit simulation using Electronic Design Automation (EDA).

```
#                          PRIMARY LoadL SERVER ==> mips27
#
# this loadl.config.local is tuned for a machine that is part of a compute
```

```
#                       SECONDARY SERVER ==> common
#
# This loadl_config.local is tuned to be "nice" to a workstation owner
# who permits loadl jobs on his system but wants good response whenever
# he is doing his own work.
#
# Run only one LoadLeveler job at a time.
#
# Check the keyboard for activity every five seconds.
#
#
```

- Batch services for serial compute jobs.
- Scheduling for parallel applications.

LoadLeveler Batch Configuration

The batch configuration is designed to maximize short job turnaround while allowing the heavy CPU jobs to get good usage of the resources available.

The basic configuration uses a range of classes – `short`, `medium`, `long` and `verylong` – with a range of maximum job CPU times of from five minutes to six days. An addition ser t èà s.

Processor Configuration

The processors are configured as follows:

- **parallel**

```
#============================================================================#
# LoadAvg is an internal variable whose value is the (Berkeley) load average
# of the machine.
#
#    CPU_Idle - No LoadL job running, or One job just finishing.
#    CPU_Busy - One LoadL job running, second job ( Foreground or Batch )
#               starting up.
#    CPU_Max  - Two LoadL jobs running.
#============================================================================#
CPU_Idle = (LoadAvg <= $(BackgroundLoad))
CPU_Busy = (LoadAvg >= $(HighLoad))

#============================================================================#
# This defines a boolean "KeyboardBusy" whose value is TRUE if the keyboard
# or mouse has been used since loadl last checked.  Thus if POLLING_FREQUENCY
# is 5 seconds, KeyboardBusy is TRUE if anybody has used the kbd or mouse in
# the last 5 seconds.
#============================================================================#
KeyboardBusy = KeyboardIdle < $(POLLING_FREQUENCY)

#============================================================================#
# This statement indicates when a job shes a— oe vcartid tnd—his sachine.
#============================================================================#
```

```
#============================================================================#
# If you set START_DAEMONS to False loadl can never start on this machine.
# For example you may want to stop loadl for a couple days for maintenance
# and make sure no procedure automatically restarts it.
#============================================================================#
START_DAEMONS = True
```

```
#============================================================================#
# Jobs in the SUSPEND state are never killed, after 6□ minutes they are
# relocated to a different box if possible.
#============================================================================#
MaxSuspendTime   = 6□ □ $(MINUTE)
VACATE    : $(StateTimer) > $(MaxSuspendTime —E)
```

```
#=============================================================================#
# Description: LoadL_config.local for Large Batch Servers ( 58□ - 59□ Class )
#=============================================================================#
# Need 3x Real Memory To Paging Space ( minimum ) For Worst Case Of One
```

Configuration for a Machine That Schedules (But Doesn't Run) Jobs

```
################################################################################
# MACHINE STANZAS:
```

Sample Configuration File

The following is a sample **LoadL_config** file.

```
#
# Machine Description
#
```

```
#
# Specify whether to start the keyboard daemon
#
X_RUNS_HERE   = True
#
#  LoadL_Startd Macros
#
STARTD   = $(BIN)/LoadL_startd
STARTD_LOG  = $(LOG)/StartLog
MAX_STARTD_LOG  = 64▫▫▫
STARTD_DEBUG  =
POLLING_FREQUENCY = 5
POLLS_PER_UPDATE─▫┐ = ┘▫REû├E┘ AQ▫NCY = 5
P┘ L_St _UPDt─LS_ô┘▫C┐─▫┐ = 5
#
STDNS_HERNCO┘▫EN┘ A┘ = LoadL_startd
```

Appendix D. Glossary

D

daemon. A process, not associated with a particular user, that performs

P

parameter

W

working directory. All files without a fully qualified path name are relative to this directory.

workstation

Index

Special Characters

/etc/LoadL.cfg file 28, 94
/etc/services file 112
.llrc script 13

A

account 78, 79
account_no 38
accounting
  API 239
  collecting data 141
  in job command file 38
  llacctmrg command 156
  llsummary command 202
  reports 143
ACCT 107
ACCT_VALIDATION 108, 240
ACTION_ON_MAX_REJECT 119
adapter
  dedicated 48
  shared 48
  specifying in job command file 47, 51
adapter information
  extracting from SDR 167
adapter stanza keywords
  adapter_name 93
  interface_address 93
  interface_name 93
  network_type 93
  switch_node_number 94
adapter stanzas
  examples 94
  format 93
adapter_name 93
adapter_stanzas 74, 75
admin keyword 82, 91
ADMIN_FILE 110
administering LoadLeveler
  administration file 69
  LoadL_admin file 72
  overview 69
  stanzas 73
administration file
  keywords 124
  sample 325
  st« b t‰re fnd b p am 72)è¢ è èà $ administrat rt 73 11)è¢ è $ AD fmŒi ltilataon) 111)è¢
  fm— e 82, 91
  sp $ ab 32 d $ A 11)è¢ è èi d 32 on Øa—d

  sp $ a)è¢ ka teEonCa—3 on Øa—d)è¢7 er–am f ltias k ,)è¢Æ

ckpt subroutine 242
class
  defining for a machine 99
  keyword 99
    in job command file 40

job *(continued)*
parallel 59, 290

log files *(continued)*
MAX_STARTER_LOG 111

OpSys
requirement in job command file 52

stanzas *(continued)*
default 73
label 73
machine 73
type 73
user 73
START expression 105
start LoadLeveler 233
START_DAEMONS 100
startd daemon 15
STARTD_DGRAM_PORT 113
STARTD_RUNS_HERE 101
STARTD_STREAM_PORT 113
startdate 54
starter process 17
state 123
states of a job 18
status 193, 200
step_name 54
stop LoadLeveler 233
submit-only machine

Communicating Your Comments to IBM

IBM LoadLeveler for AIX
Using and Administering
Version 2 Release 1

Publication No. SA22-7311-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you

Readers' Comments — We'd Like to Hear from You

IBM LoadLeveler for AIX
Using and Administering
Version 2 Release 1

Publication No. SA22-7311-00

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note:

IBM

Program Number: 5765-D61