

DIETBatchManagementAPI

Generated by Doxygen 1.7.2

Thu Nov 11 2010 11:54:55

Contents

1	The diet-bms-lib Index Page	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	job.t Struct Reference	7
4.1.1	Detailed Description	8
4.1.2	Member Data Documentation	8
4.1.2.1	id	8
4.1.2.2	jobClass	8
4.1.2.3	name	8
4.1.2.4	nbNodes	8
4.1.2.5	priority	8
4.1.2.6	status	8
4.1.2.7	subTime	8
4.1.2.8	user	8
4.1.2.9	walltime	9
4.2	listOfJobs.t Struct Reference	9
4.2.1	Detailed Description	9
4.2.2	Member Data Documentation	9
4.2.2.1	jobs	9
4.2.2.2	nbJobs	10
4.2.2.3	nblots	10
4.2.2.4	nbRunningJobs	10
4.2.2.5	nbWaitingJobs	10
5	File Documentation	11
5.1	diet-bmms-lib.h File Reference	11
5.1.1	Detailed Description	12
5.1.2	Function Documentation	12
5.1.2.1	dietBuildLot	12
5.1.2.2	dietCancelLot	13
5.1.2.3	dietGetOutPutLot	13
5.1.2.4	dietListLot	14
5.1.2.5	dietStatusLot	15
5.1.2.6	dietSubmitLot	15

5.2	diet-bms-lib.h File Reference	16
5.2.1	Detailed Description	17
5.2.2	Function Documentation	17
5.2.2.1	dietcancel	17
5.2.2.2	dietlist	18
5.2.2.3	dietNbJobs	19
5.2.2.4	dietNbRunningJobs	19
5.2.2.5	dietNbWaitingJobs	20
5.2.2.6	dietsubmit	21
5.2.2.7	printJob	22
5.2.2.8	printListOfJobs	22
5.3	JobStructures.h File Reference	22
5.3.1	Detailed Description	22
5.3.2	List of jobs structure	22
5.3.3	Job structure	23

Chapter 1

The diet-bms-lib Index Page

This documentation is intended to provide a description of the API for the DIET Batch Management System. There is two files to look at:

- [diet-bms-lib.h](#) : it describes the main functions of the API
- [diet-bmms-lib.h](#) : it describes the advanced functions of the API concerning the multi-jobs features
- [JobStructures.h](#) : it contains the structures you have to use to get information on jobs

Version

1.1

Author

GRAAL, INRIA Rhone-Alpes

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

job.t (Structure of a job)	7
listOfJobs.t (Structure of a list of jobs)	9

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

diet-bmms-lib.h (This file can be used to access the API for the DIET Batch Management Multi-jobs System)	11
diet-bms-lib.h (This file can be used to access the API for the DIET Batch Management System)	16
JobStructures.h (This file stores the two structures that are used to interact with list of jobs and jobs)	22

Chapter 4

Class Documentation

4.1 job_t Struct Reference

Structure of a job.

```
#include <JobStructures.h>
```

Public Attributes

- char * [id](#)
Id of the job.
- char * [name](#)
Name of the job.
- char * [user](#)
Owner of the job.
- char * [subTime](#)
Submission time.
- char * [status](#)
Status of the job.
- char * [priority](#)
Priority of the job.
- char * [jobClass](#)
Class of the job.
- char * [walltime](#)
Walltime of the job.

- char * [nbNodes](#)

Number of nodes of the job.

4.1.1 Detailed Description

Structure of a job.

4.1.2 Member Data Documentation

4.1.2.1 char* job_t::id

Id of the job.

4.1.2.2 char* job_t::jobClass

Class of the job.

4.1.2.3 char* job_t::name

Name of the job.

4.1.2.4 char* job_t::nbNodes

Number of nodes of the job.

4.1.2.5 char* job_t::priority

Priority of the job.

4.1.2.6 char* job_t::status

Status of the job.

4.1.2.7 char* job_t::subTime

Submission time.

4.1.2.8 char* job_t::user

Owner of the job.

4.1.2.9 char* job_t::walltime

Walltime of the job.

The documentation for this struct was generated from the following file:

- [JobStructures.h](#)

4.2 listOfJobs_t Struct Reference

Structure of a list of jobs.

```
#include <JobStructures.h>
```

Public Attributes

- unsigned long long [nbJobs](#)
Total number of jobs.
- unsigned long long [nblots](#)
Total number of lots.
- unsigned long long [nbRunningJobs](#)
Number of running jobs.
- unsigned long long [nbWaitingJobs](#)
Number of waiting jobs.
- [job_t](#) * [jobs](#)
Array containing the jobs.

4.2.1 Detailed Description

Structure of a list of jobs.

4.2.2 Member Data Documentation

4.2.2.1 job_t* listOfJobs_t::jobs

Array containing the jobs.

4.2.2.2 unsigned long long listOfJobs_t::nbJobs

Total number of jobs.

Note

this is the size of the jobs array

4.2.2.3 unsigned long long listOfJobs_t::nbLots

Total number of lots.

Note

this is the size of the jobs array

4.2.2.4 unsigned long long listOfJobs_t::nbRunningJobs

Number of running jobs.

4.2.2.5 unsigned long long listOfJobs_t::nbWaitingJobs

Number of waiting jobs.

The documentation for this struct was generated from the following file:

- [JobStructures.h](#)

Chapter 5

File Documentation

5.1 diet-bmms-lib.h File Reference

This file can be used to access the API for the DIET Batch Management Multi-jobs System.

```
#include <sys/types.h>
```

Functions

- char * [dietSubmitLot](#) (const char *path_to_lot, char **errorMessage, char **hostname)

Function used to submit a lot (a set of jobs) on a machine.

- char * [dietBuildLot](#) (const char *hostname, int nbIDs, const char *const *listIDs, char **errorMessage)

Function used to build a lot (a set of jobs) with given ids on a machine.

- char * [dietListLot](#) (const char *hostname)

Function used to get the information of all lots on a machine.

- char * [dietStatusLot](#) (const char *hostname, const char *ID)

Function used to get the status of a lot (a set of jobs) on a machine.

- char * [dietCancelLot](#) (const char *hostname, const char *ID)

Function used to cancel a lot (a set of jobs) of a given id on a machine.

- char * [dietGetOutPutLot](#) (const char *hostname, const char *ID, char **output, char **error, char **output_path, char **error_path)

Function used to get the output of a lot (a set of jobs) on a machine.

5.1.1 Detailed Description

This file can be used to access the API for the DIET Batch Management Multi-jobs System.

5.1.2 Function Documentation

5.1.2.1 `char* dietBuildLot (const char * hostname, int nbIDs, const char *const * listIDs, char ** errorMessage)`

Function used to build a lot (a set of jobs) with given ids on a machine.

Parameters

<i>hostname</i>	name of the machine
<i>number</i>	of IDs
<i>listIDs</i>	the list of ids that you want to build a lot with
<i>errorMessage</i>	error message

Returns

the ID of the lot

Here is a example code using the dietBuildLot function to create a lot out of a list of job IDs on a given machine.

```
#include <stdio.h>
#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bmms-lib.h"

int main(int argc, char* argv[]){

    diet_initialize(argv[1], argc, argv);

    char **listIDs = (char**) malloc(sizeof(char*) * (argc - 3));
    for(int i=3; i < argc; i++)
        listIDs[i] = argv[i];

    char *errorMessage = NULL;
    char *lotID = NULL;
    lotID = dietBuildLot("fen", listIDs, &errorMessage);

    if(strlen(job_id)!=0)
        printf("Lot ID: %s\n", lotID);
    else
        printf("Error: %s\n", errorMessage);

    free(errorMessage);
    free(lotID);
    free(listIDs);
    diet_finalize();
}
```


5.1.2.2 `char* dietCancelLot (const char * hostname, const char * ID)`

Function used to cancel a lot (a set of jobs) of a given id on a machine.

Parameters

<i>hostname</i>	name of the machine
<i>ID</i>	identifiant of the lot or job

Returns

execution message

Here is a example code using the dietCancelLot function to cancel a lot on a given machine.

```
#include <stdio.h>
#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bmms-lib.h"

int main(int argc, char* argv[]){

    diet_initialize(argv[1], argc, argv);

    char *ID = argv[2];
    char *message = NULL;

    message = dietCancelLot("fen", ID);

    printf("%s\n", message);
    free(message);
    diet_finalize();
}
```

5.1.2.3 `char* dietGetOutPutLot (const char * hostname, const char * ID, char ** output, char ** error, char ** output_path, char ** error_path)`

Function used to get the output of a lot (a set of jobs) on a machine.

Parameters

<i>hostname</i>	name of the machine
<i>ID</i>	identifiant of the lot or job
<i>output</i>	the output information
<i>error</i>	the error information
<i>output_path</i>	the output path specified in the script
<i>error_path</i>	the error path specified in the script

Returns

execution message

Here is a example code using the `dietGetOutputLot` function to retrieve the the lot stdout and stderr of a job.

```
#include <stdio.h>
#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bmms-lib.h"

int main(int argc, char* argv[]){

    diet_initialize(argv[1], argc, argv);

    char *ID = argv[2];
    char *output = NULL;
    char *error = NULL;
    char *output_path = NULL;
    char *error_path = NULL;
    char *cmd_output = NULL;
    cmd_output = dietGetOutputLot("fen", ID, &output, &error,
                                &output_path, &error_path);

    printf("The output of the job id %s is: %s\n", ID, output);
    printf("The output path of the job id %s is: %s\n", ID, output_path);
    printf("The error of the job id %s is: %s\n", ID, error);
    printf("The error path of the job id %s is: %s\n", ID, error_path);
    printf("Execution message is: %s\n", cmd_output);
    free(output);
    free(error);
    free(output_path);
    free(error_path);
    free(cmd_output);
    diet_finalize();
}
```

5.1.2.4 `char* dietListLot (const char * hostname)`

Function used to get the information of all lots on a machine.

Parameters

<i>hostname</i>	name of the machine
-----------------	---------------------

Returns

the listing informations

Here is a example code using the `dietListLot` function to list all lots present on a given machine.

```
#include <stdio.h>
#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bmms-lib.h"

int main(int argc, char* argv[]){
```

```

diet_initialize(argv[1], argc, argv);
char * listInfo = dietListLot("fen");

printf("Info: %s\n", listInfo);
free(listInfo);
diet_finalize();
}

```

5.1.2.5 char* dietStatusLot (const char * *hostname*, const char * *ID*)

Function used to get the status of a lot (a set of jobs) on a machine.

Parameters

<i>hostname</i>	name of the machine
<i>ID</i>	identifiant of the lot or job

Returns

the status informations

Here is a example code using the dietStatusLot function to get the status of a lot on a given machine.

```

#include <stdio.h>
#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bmms-lib.h"

int main(int argc, char* argv[]){

    diet_initialize(argv[1], argc, argv);

    char *ID = argv[2];
    char *status = NULL;

    status = dietStatusLot("fen", ID);

    printf("Status: %s\n", status);
    free(status);
    diet_finalize();
}

```

5.1.2.6 char* dietSubmitLot (const char * *path_to_lot*, char ** *errorMessage*, char ** *hostname*)

Function used to submit a lot (a set of jobs) on a machine.

Parameters

<i>path_to_lot</i>	the path to get the lot file or script file
<i>errorMessage</i>	error message
<i>hostname</i>	the host on which the job has been submitted

Returns

the ID of the lot

Here is a example code using the `dietSubmitLot` function to submit the lot on a best selected machine.

```
#include <stdio.h>
#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bmms-lib.h"

int main(int argc, char* argv[]){

    diet_initialize(argv[1], argc, argv);

    char *my_lot = argv[2];
    char *errorMessage = NULL;
    char *hostname = NULL;
    char *lotID = NULL;

    lotID = dietSubmitLot(my_lot, &errorMessage, &hostname);

    if(strlen(job_id)!= 0 && strlen(hostname) != 0)
        printf("Lot submitted on %s, with ID: %s\n", hostname, lotID);
    else
        printf("Error: %s\n", errorMessage);

    free(errorMessage);
    free(hostname);
    free(lotID);
    diet_finalize();
}
```

5.2 diet-bms-lib.h File Reference

This file can be used to access the API for the DIET Batch Management System.

```
#include <sys/types.h>
#include "JobStructures.h"
```

Functions

- `int dietlist` (const char *hostname, [listOfJobs_t](#) *listOfJobs, char *errorMessage, const size_t errorSize)
Function used to list the jobs on a machine.
- `int dietNbJobs` (const char *hostname, char *errorMessage, const size_t errorSize)
Function used to get the number of jobs on a machine.
- `int dietNbRunningJobs` (const char *hostname, char *errorMessage, const size_t errorSize)

Function used to get the number of running jobs on a machine.

- int [dietNbWaitingJobs](#) (const char *hostname, char *errorMessage, const size_t errorSize)

Function used to get the number of waiting jobs on a machine.

- int [dietcancel](#) (const char *hostname, const char *jobId, char *outputMessage, const size_t maxsize, char *errorMessage, const size_t errorSize)

Function used to cancel a job on a machine with its job id.

- int [dietsubmit](#) (const char *hostname, const char *scriptPath, char *jobID, const size_t maxsize, char *error_message, const size_t error_maxsize)

Function used to submit a job on a machine with a script.

- int [printListOfJobs](#) (const [listOfJobs_t](#) *listOfJobs)

Method used to the a list of jobs.

- int [printJob](#) (const [job_t](#) *job)

Method used to print a job.

5.2.1 Detailed Description

This file can be used to access the API for the DIET Batch Management System.

5.2.2 Function Documentation

5.2.2.1 int [dietcancel](#) (const char * *hostname*, const char * *jobId*, char * *outputMessage*, const size_t *maxsize*, char * *errorMessage*, const size_t *errorSize*)

Function used to cancel a job on a machine with its job id.

Parameters

<i>hostname</i>	the name of the machine
<i>jobId</i>	the id of the job to cancel
<i>outputMessage</i>	message delivered by the scheduler
<i>maxsize</i>	the maximum length of the buffer where the output message will be stored
<i>errorMessage</i>	error message
<i>errorSize</i>	size of the error message

Here is an example of code using the [dietcancel](#) function to cancel a job with ID "bar" on the target machine called "foo":

```
#include <stdio.h>
```

```

#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bms-lib.h"

int main(int argc, char* argv[]){

    diet_initialize(argv[1], argc, argv);

    char* outputMessage = (char*)malloc(255*sizeof(char));
    char* errorMessage = (char*)malloc(1000*sizeof(char));

    dietcancel("foo", "bar", outputMessage, 255, errorMessage, 1000);

    printf("%s\n", outputMessage);

    free(outputMessage);
    free(errorMessage);

    diet_finalize();

}

```

5.2.2.2 int dietlist (const char * *hostname*, listOfJobs_t * *listOfJobs*, char * *errorMessage*, const size_t *errorSize*)

Function used to list the jobs on a machine.

Parameters

<i>hostname</i>	name of the machine
<i>listOfJobs</i>	structure that will at the end contain the list of jobs
<i>errorMessage</i>	error message
<i>errorSize</i>	size of the error message

Here is a example code using the dietlist function to retrieve the existing jobs on a machine called "foo":

```

#include <stdio.h>
#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bms-lib.h"

int main(int argc, char* argv[]){

    diet_initialize(argv[1], argc, argv);

    listOfJobs_t listOfJobs;
    char* errorMessage = (char*)malloc(1000*sizeof(char));

    dietlist("foo", &listOfJobs, errorMessage, 1000);

    printListOfJobs(&listOfJobs);

    free(errorMessage);
}

```

```
diet_finalize();

}
```

5.2.2.3 int dietNbJobs (const char * *hostname*, char * *errorMessage*, const size_t *errorSize*)

Function used to get the number of jobs on a machine.

Parameters

<i>hostname</i>	name of the machine
<i>errorMes-</i> <i>sage</i>	error message
<i>errorSize</i>	size of the error message

Returns

the number of jobs

Here is an example of code using the dietNbJobs function to retrieve the number of jobs on the target machine called "foo":

```
#include <stdio.h>
#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bms-lib.h"

int main(int argc, char* argv[]){

    diet_initialize(argv[1], argc, argv);

    char* errorMessage = (char*)malloc(1000*sizeof(char));

    int nbJobs = dietNbJobs("Arrakis-2.local", errorMessage, 1000);

    printf("Total number of jobs on %s is %d\n","Arrakis-2.local",nbJobs);

    free(errorMessage);

    diet_finalize();

}
```

5.2.2.4 int dietNbRunningJobs (const char * *hostname*, char * *errorMessage*, const size_t *errorSize*)

Function used to get the number of running jobs on a machine.

Parameters

<i>hostname</i>	name of the machine
<i>errorMes-</i> <i>sage</i>	error message
<i>errorSize</i>	size of the error message

Returns

the number of running jobs

Here is an example of code using the `dietNbRunningJobs` function to retrieve the number of running jobs on the target machine called "foo":

```
#include <stdio.h>
#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bms-lib.h"

int main(int argc, char* argv[]){

    diet_initialize(argv[1], argc, argv);

    char* errorMessage = (char*)malloc(1000*sizeof(char));

    int nbRunningJobs = dietNbRunningJobs("foo", errorMessage, 1000);

    printf("Total number of running jobs on %s is %d\n", "foo", nbRunningJobs);

    free(errorMessage);

    diet_finalize();

}
```

5.2.2.5 int dietNbWaitingJobs (const char * *hostname*, char * *errorMessage*, const size_t *errorSize*)

Function used to get the number of waiting jobs on a machine.

Parameters

<i>hostname</i>	name of the machine
<i>errorMessage</i>	error message
<i>errorSize</i>	size of the error message

Returns

the number of waiting jobs

Here is an example of code using the `dietNbWaitingJobs` function to retrieve the number of waiting jobs on the target machine called "foo":

```
#include <stdio.h>
#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bms-lib.h"

int main(int argc, char* argv[]){

    diet_initialize(argv[1], argc, argv);
```



```

char* errorMessage = (char*) malloc(1000*sizeof(char));

int nbWaitingJobs = dietNbWaitingJobs("foo");

printf("Total number of waiting jobs on %s is %d\n", "foo", nbWaitingJobs);

free(errorMessage);

diet_finalize();

}

```

5.2.2.6 int dietsubmit(const char * *hostname*, const char * *scriptPath*, char * *jobID*, const size_t *maxsize*, char * *error_message*, const size_t *error_maxsize*)

Function used to submit a job on a machine with a script.

Parameters

<i>hostname</i>	name of the machine
<i>scriptPath</i>	path of the script to submit
<i>jobID</i>	id of the job that has been submitted
<i>maxsize</i>	maximum size of data that should be stored in the jobID variable
<i>error_message</i>	message containing the possible error message thrown during the submission
<i>error_maxsize</i>	size of the data that should be stored in the error_message variable

Here is an example of code using the dietsubmit function to submit a job with scrip file bar.txt on the target machine called "foo":

```

#include <stdio.h>
#include <stdlib.h>
#include "JobStructures.h"
#include "diet-bms-lib.h"

int main(int argc, char* argv[]){

diet_initialize(argv[1], argc, argv);

char* jobID = (char*)malloc(255*sizeof(char));

char* errorMessage = (char*)malloc(1000*sizeof(char));

dietsubmit("foo", "bat.txt", jobID, 255, errorMessage, 1000);

printf("%s\n", jobID);

free(jobID);

diet_finalize();

}

```

5.2.2.7 `int printJob (const job_t * job)`

Method used to print a job.

Parameters

<i>job</i>	a job to pring
------------	----------------

5.2.2.8 `int printListOfJobs (const listOfJobs_t * listOfJobs)`

Method used to the a list of jobs.

Parameters

<i>listOfJobs</i>	a list of jobs to print
-------------------	-------------------------

5.3 JobStructures.h File Reference

This file stores the two structures that are used to interact with list of jobs and jobs.

Classes

- struct `job_t`
Structure of a job.
- struct `listOfJobs_t`
Structure of a list of jobs.

5.3.1 Detailed Description

This file stores the two structures that are used to interact with list of jobs and jobs.

5.3.2 List of jobs structure

The list of jobs is a simple array of jobs with three values added that are

- `nbJobs` which is the total number of jobs contained in the list
- `nbRunningJobs` which is the number of running jobs in the list
- `nbWaitingJobs` which is the number of waiting jobs in the list

Note

the `nbJobs` is the size of the jobs array

5.3.3 Job structure

the job structure is a simple job description

Version

1.1

Author

GRAAL, INRIA Rhones-Alpes

Index

diet-bmms-lib.h, 11
 dietBuildLot, 12
 dietCancelLot, 12
 dietGetOutPutLot, 13
 dietListLot, 14
 dietStatusLot, 15
 dietSubmitLot, 15
diet-bms-lib.h, 16
 dietcancel, 17
 dietlist, 18
 dietNbJobs, 19
 dietNbRunningJobs, 19
 dietNbWaitingJobs, 20
 dietsubmit, 21
 printJob, 21
 printListOfJobs, 22
dietBuildLot
 diet-bmms-lib.h, 12
dietcancel
 diet-bms-lib.h, 17
dietCancelLot
 diet-bmms-lib.h, 12
dietGetOutPutLot
 diet-bmms-lib.h, 13
dietlist
 diet-bms-lib.h, 18
dietListLot
 diet-bmms-lib.h, 14
dietNbJobs
 diet-bms-lib.h, 19
dietNbRunningJobs
 diet-bms-lib.h, 19
dietNbWaitingJobs
 diet-bms-lib.h, 20
dietStatusLot
 diet-bmms-lib.h, 15
dietsubmit
 diet-bms-lib.h, 21
dietSubmitLot
 diet-bmms-lib.h, 15

id
 job_t, 8

job_t, 7
 id, 8
 jobClass, 8
 name, 8
 nbNodes, 8
 priority, 8
 status, 8
 subTime, 8
 user, 8
 walltime, 8

jobClass
 job_t, 8

jobs
 listOfJobs_t, 9
JobStructures.h, 22

listOfJobs_t, 9
 jobs, 9
 nbJobs, 9
 nblots, 10
 nbRunningJobs, 10
 nbWaitingJobs, 10

name
 job_t, 8
nbJobs
 listOfJobs_t, 9
nblots
 listOfJobs_t, 10
nbNodes
 job_t, 8
nbRunningJobs
 listOfJobs_t, 10
nbWaitingJobs
 listOfJobs_t, 10

printJob
 diet-bms-lib.h, 21
printListOfJobs

diet-bms-lib.h, [22](#)

priority
 job_t, [8](#)

status
 job_t, [8](#)

subTime
 job_t, [8](#)

user
 job_t, [8](#)

walltime
 job_t, [8](#)