FUJITSU

FUJITSU Software
Technical Computing Suite V4.0L20

ジョブ運用ソフトウェア
APIユーザーズガイド コマンドAPI編

J2UL-2544-01Z0(01)
2020年9月

まえがき

本書の目的

本書では、Technical Computing Suiteのジョブ運用ソフトウェアが提供する、ジョブ運用管理機能のコマンドAPIについて説明します。

本書の読者

本書は、ジョブ運用ソフトウェアによるジョブの運用と管理をする管理者と、実際にジョブを操作するエンドユーザが対象です。

本書を読むためには、以下の知識が必要です。

- Linux に関する基本的な知識
- 「ジョブ運用ソフトウェア エンドユーザ向けガイド」による、ジョブの操作(投入や削除など)の知識
- 「ジョブ運用ソフトウェア 管理者向けガイド ジョブ管理編」による、ジョブ運用に関する操作の知識

本書の構成

本書は、次の構成になっています。

第1章 コマンドAPIの概要

コマンドAPIの概要について説明します。

第2章 コマンドAPIの利用

コマンドAPIの利用方法について説明します。

付録A リファレンス:コマンドAPI共通

コマンドAPIに共通する操作や情報参照のためのAPIを説明します。

付録B リファレンス:ジョブ操作API

ジョブの操作をするためのAPIについて説明します。

付録C リファレンス:情報取得API

ジョブや資源の情報を取得するためのAPIついて説明します。

付録D リファレンス:ジョブ運用の設定操作API

ジョブ投入や実行を許可ためのAPIについて説明します。

付録E サンプルプログラム

コマンドAPIを使った簡単なサンプルプログラムです。

本書の表記について

単位の表現

本書では、単位を表現する際の接頭語は以下のとおりです。基本的にディスクサイズは10のべき乗、メモリサイズは2のべき乗で表現します。コマンドの表示や入力時の指定の際は注意してください。

接頭語	値	接頭語	値
K (kilo)	10^3	Ki (kibi)	2^{10}
M (mega)	10^6	Mi (mebi)	2^{20}
G (giga)	10^9	Gi (gibi)	2^{30}
T (tera)	10^{12}	Ti (tebi)	2^{40}
P (peta)	10^{15}	Pi (pebi)	2^{50}

機種名の表現

本書では富士通製CPU A64FXを搭載した計算機を「FXサーバ」、FUJITSU server PRIMERGYを「PRIMERGYサーバ」(または単に「PRIMERGY」)と表記します。
また、本書で説明する機能の一部には、対象機種によって仕様に差があります。このような機能の説明では、以下のように対象機種を略称で表記します。
[FX] : FXサーバを対象にした機能です。
[PG] : PRIMERGYサーバを対象にした機能です。

コマンドのパス名の表記

操作例で、ディレクトリ/bin、/usr/bin、/sbin、または/usr/sbin配下にあるコマンドについては絶対パスで示していない場合があります。

マニュアル内のアイコンについて

本書では、以下のアイコンを使用しています。

注意

特に注意が必要な事項を説明しています。必ずお読みください。

参照

詳細な情報が書かれている参照先を示しています。

参考

ジョブ運用ソフトウェアに関連した参考記事を説明しています。

輸出管理規制について

本ドキュメントを輸出または第三者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ、必要な手続きをおとりください。

商標

- Linux®は米国及びその他の国におけるLinus Torvaldsの登録商標です。
- Red Hat、Red Hat Enterprise Linuxは米国およびその他の国において登録されたRed Hat, Inc.の商標です。
- そのほか、本マニュアルに記載されている会社名および製品名は、それぞれ各社の商標または登録商標です。

出版年月および版数

版数	マニュアルコード
2020年9月 第1.1版	J2UL-2544-01Z0(01)
2020年2月 初版	J2UL-2544-01Z0(00)

著作権表示

変更履歴

変更内容	変更箇所	版数
関数pjcmd_submit_put_job_resource()のパラメーター"node"で指定できる値に、"no-io-exclusive"と"io-exclusive"を追加しました。	B.1.10	第1.1版

目　次

第1章 コマンドAPIの概要

本章では、コマンドAPIの概要を説明します。

1.1 コマンドAPIとは

ジョブ運用で求められるユーザーインターフェースは運用システムごとに様々です。
例えば、ジョブ運用の方針に合わせてコマンドのオプション名を変更したり、無効化したりしたい場合があります。また、コマンドのメッセージやコマンドが出力する情報を独自の書式に変更したい場合もあります。

利用者が望むユーザーインターフェースを持ったコマンドの作成を支援するために、ジョブ運用管理機能はそれが提供するコマンドと同等の機能(ジョブの操作や情報取得)をプログラムから呼び出すためのインターフェースを提供します。これをコマンドAPI(Application Programming Interface)と呼びます。これにより、管理者やエンドユーザが望むユーザーインターフェースを持ったコマンドを作成できます。

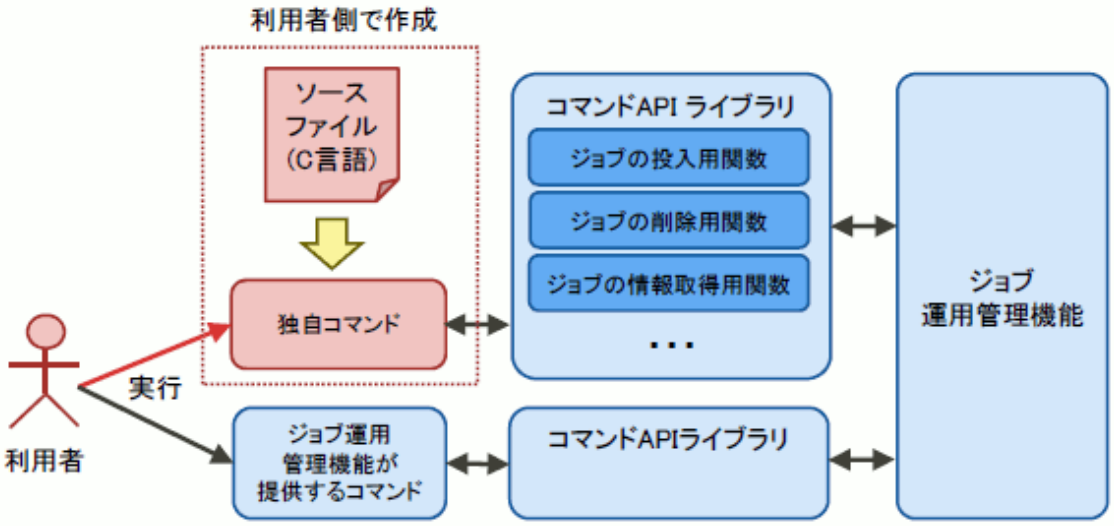
コマンドAPIはC言語の関数群で、ライブラリとして提供されます。このライブラリはログインノード、計算クラスタ管理ノード、およびシステム管理ノードで利用できます。

コマンドAPIがサポートしている機能は以下です。

表1.1 コマンドAPIがサポートしている機能

機能	機能に相当するコマンド
ジョブの投入	pjsubコマンド
ジョブの削除	pjdelコマンド
ジョブのパラメーター変更	pjalterコマンド、pmalterコマンド
ジョブへのシグナル送信	pjsigコマンド
ジョブの固定、固定解除	pjholdコマンド、pjrlsコマンド
ジョブの終了待ち合わせ	pjwaitコマンド
ジョブの情報取得	pjstatコマンド (-s、-Sオプション)
リソースユニットやリソースグループの情報取得	pjstatコマンド (--rscオプション)
ジョブの制限値の情報取得	pjstatコマンド (--limitオプション)
ジョブACL機能の情報取得	pjaclコマンド
ジョブの資源利用状況の取得	pjshowrscコマンド
ジョブ運用の変更およびジョブ運用の状態取得 (ジョブの投入・実行の許可の設定および設定状況の取得)	pmpjmoptコマンド (--set-rsc-ruオプション、--show-rsc-ugオプション)

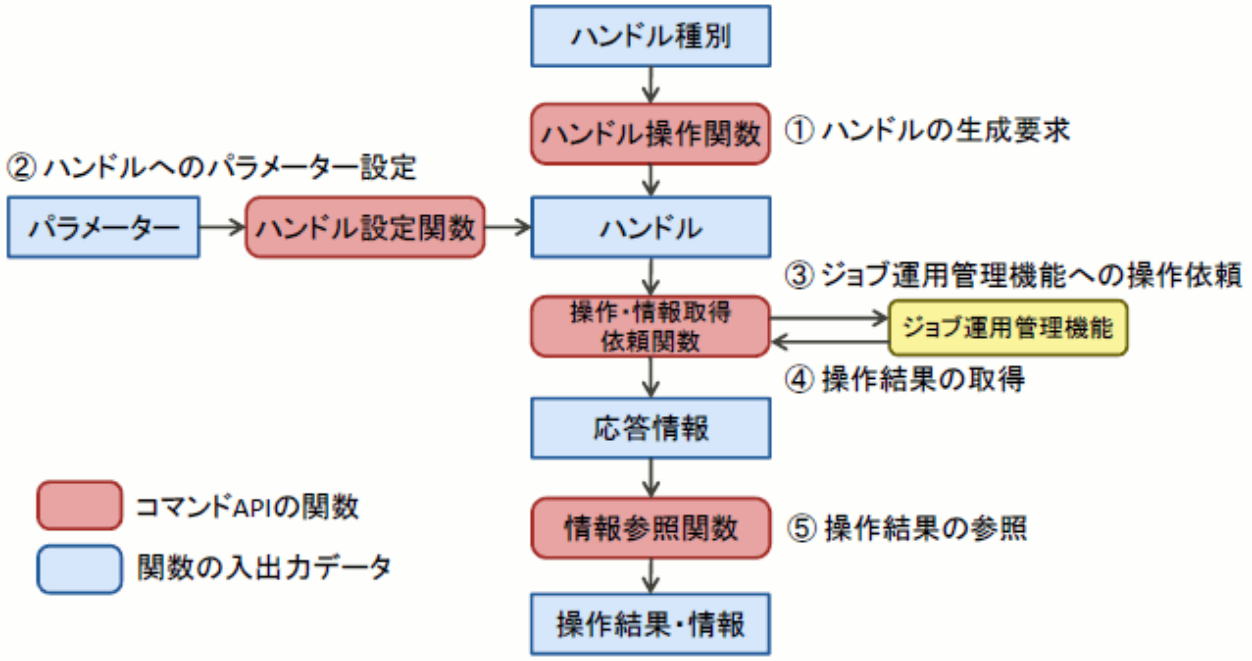
図1.1 利用イメージ



1.2 操作フロー

コマンドAPIによる操作は、以下に示すフローになります。具体的な利用方法は"第2章 コマンドAPIの利用"を参照してください。

図1.2 コマンドAPIによる操作フロー



1. ハンドルの生成要求

   コマンドAPIを利用するには、まずハンドル操作関数を使ってハンドルを生成します。ハンドルとは、ジョブの操作や情報取得をジョブ運用管理機能へ依頼するためのパラメーターを保持する情報です。ジョブ運用管理機能への依頼の種類に応じて、生成するハンドルの種類を指定します。

2. ハンドルへのパラメーター設定

   生成したハンドルに対して、ハンドル設定関数でパラメーターを設定します。パラメーターとは、投入するジョブに割り当てる資源量や削除対象のジョブIDなど、ジョブ運用管理機能への依頼の内容を示す情報です。

3. ジョブ運用管理機能への操作依頼

   パラメーターをハンドルに設定したら、依頼関数でジョブ運用管理機能へ操作を依頼します。ジョブ運用管理機能はハンドルの内容に基づいて処理をします。

4. 操作結果の取得

   ジョブ運用管理機能への依頼の結果である応答情報が、依頼関数の戻り値として返されます。

5. 操作結果の参照

   応答情報には、操作の成否や取得した情報が含まれます。これらの情報は情報参照関数で参照します。情報取得の場合は、応答情報からさらに詳細な情報を含む構造体を取得し、参照する場合もあります。

1.3 関数の種類

コマンドAPIの関数は、操作の種類ごとに以下のような名前の関数があります。

表1.2 コマンドAPIの関数

操作	関数名
ジョブの投入	pjcmd_submit_*action*()
ジョブの削除	pjcmd_kill_*action*()

操作	関数名
ジョブのパラメーター変更	pjcmd_alter_*action*()
ジョブへのシグナル送信	pjcmd_signal_*action*()
ジョブの固定	pjcmd_hold_*action*()
ジョブの固定解除	pjcmd_release_*action*()
ジョブの終了待ち合わせ	pjcmd_wait_*action*()
ジョブの情報取得	pjcmd_jobinfo_*action*()
リソースユニットやリソースグループの情報取得	pjcmd_rscinfo_*action*()
ジョブ投入時の制限値の情報取得	pjcmd_limitinfo_*action*()
ジョブACL機能の情報取得	pjcmd_jacl_*action*()
ジョブの資源利用状況の取得	pjcmd_rscstat_*action*()
ジョブ運用の変更およびジョブ運用の状態取得 (ジョブの投入・実行の許可の設定および設定状況の取得)	pjcmd_setpjmstat_*action*() pjcmd_getpjmstat_*action*()
ユーティリティ関数 (ハンドルや応答情報の操作、結果を参照するなど、コマンドAPIの利用における補助的な関数)	pjcmd_*action*()

関数名の "*action*" 部分は処理の種類を示します。処理には以下があります。

表1.3 コマンドAPI関数の処理の種類

処理	説明	関数の例
ハンドル操作	ハンドルの生成、初期化、複製、および解放をする関数です。 それぞれの操作に対して関数が用意されています。	pjcmd_create_handle() pjcmd_reset_handle() pjcmd_clone_handle() pjcmd_destroy_handle()
パラメーター設定と参照	ジョブの操作や情報取得などのパラメーターをハンドルに設定する関数および設定されている値を参照する関数です。 機能の種類ごと、パラメーターの種類ごとに関数があります。	pjcmd_submit_put_param() pjcmd_submit_get_param()
引数解析	与えられたコマンドライン引数をジョブ運用管理機能のコマンドのオプション仕様に従って解析し、ハンドルにパラメーターを設定します。 コマンドの引数を利用者側で解析するのではなく、ジョブ運用ソフトウェアのコマンドのオプション体系に従ってコマンドAPIが解析し、ハンドルにパラメーターを設定させる場合に利用します。 また、pjsubコマンド相当のオプションについてはパーサと呼ぶ機能でカスタマイズ(オプション名の変更と無効化)ができます。	pjcmd_submit_parse_pjsub_args() pjcmd_getopt_long() pjcmd_renameopt_in_parser() pjcmd_delopt_in_parser()
操作要求	ハンドルに設定された情報に基づいて、ジョブ運用管理機能に対して、ジョブの操作や情報取得を要求します。機能の種類ごとに関数があります。	pjcmd_submit_execute()
結果参照	操作や情報取得の結果である応答情報から、依頼の成否や取得した情報(ジョブIDやジョブ情報など)を参照します。 操作の種類や参照する情報ごとに関数があります。	pjcmd_get_jobresult_info()
表示	応答情報の内容を、標準コマンドの表示仕様に基づいて出力します。 情報の表示をカスタマイズする必要がない場合に利用できます。	pjcmd_jobinfo_print_resp()
応答情報操作	応答情報を解放します。 応答情報はコマンドAPI内で確保されるため、必要がなくなれば利用側で解放します。	pjcmd_destroy_resp()

参照

関数の詳細は、"付録A リファレンス:コマンドAPI共通"から"付録D リファレンス:ジョブ運用の設定操作API"を参照してください。

第2章 コマンドAPIの利用

本章では、コマンドAPIの利用方法について説明します。

2.1 ヘッダーファイル

コマンドAPIのヘッダーファイルは、ログインノード、計算クラスタ管理ノード、およびシステム管理ノードの以下にあります。

```
/usr/include/FJSVtcs/pjm/pjcmd.h
```

コマンドAPIを使うソースファイルでは、このヘッダーファイルをインクルードしてください。

```
#include <FJSVtcs/pjm/pjcmd.h>
...
main()
{
    // コマンドAPIを利用した処理
    ...
}
```

2.2 コマンドAPIによる操作

ここでは、コマンドAPIによる操作フローに沿って、代表的な関数の利用方法を説明します。

2.2.1 ハンドルの生成

コマンドAPIでは、操作に関するパラメーターをハンドルに設定し、ジョブ運用管理機能へ依頼します。このため、最初にハンドルを作成する必要があります。

ハンドルは、関数pjcmd_create_handle()で作成します。

```
PjcmdHandle_t *handle_p;

handle_p = pjcmd_create_handle(PJCMD_SUBMIT);  // ジョブ投入用のハンドルの生成
```

関数pjcmd_create_handle()は、ハンドル(PjcmdHandle_t)を作成し、成功した場合はその領域へのポインターを返します。

注意

ハンドルは、操作の種類ごとに用意します。例えば、ジョブ投入用に生成したハンドルは、ジョブ削除用には使えません。

2.2.2 パラメーターの設定

ハンドルには、操作の内容を表すパラメーターを設定します。パラメーターはジョブ運用管理機能のコマンド(pjsubコマンドなど)のオプションに相当するものです。

例えば、バルクジョブを投入する場合、pjsubコマンドでは以下のように引数を指定します。

```
pjsub -L "node=8x8" --bulk --sparam "1-10" ./job.sh
```

これをコマンドAPIで実現する場合、これらのオプションに相当するパラメーターは、以下のようにハンドルに設定します。

```
char *jobscript_p = "./job.sh";                    // ジョブスクリプト ./job.sh
char *node_p = "8x8";                              // ノード形状 8x8
pjcmd_jobmodel_t jobmodel = PJCMD_JOBMODEL_BULK; // ジョブモデル: バルクジョブ
uint32_t bulk_sno = 1, bulk_eno = 10;             // バルク開始番号1、バルク終了番号10

pjcmd_submit_put_param(handle_p, PJCMD_SUBMIT_SCRIPTFILE, &jobscript_p);
pjcmd_submit_put_job_resource(handle_p, "node", &node_p);
```

```
pjcmd_submit_put_param(handle_p, PJCMD_SUBMIT_JOBMODEL, &jobmodel);
pjcmd_submit_put_param(handle_p, PJCMD_SUBMIT_BULK_STARTNO, &bulk_sno);
pjcmd_submit_put_param(handle_p, PJCMD_SUBMIT_BULK_ENDNO, &bulk_eno);

※関数の復帰値の判定は省略しています。
```

注意

- 上記ではハンドルにパラメーターの値(ポインターや数値)を設定するとき、パラメーターの値ではなく、パラメーターの値が格納された領域のアドレスを渡していることに注意してください。これは、1つの関数でいろいろなデータ型のパラメーターを扱うためです。例えば、関数pjcmd_submit_put_param()では、第2引数でパラメーターの種類を指定し、第3引数(void *型)でパラメーターが格納された領域のアドレスを指定します。関数は第2引数で示されるパラメーターの種類によって、パラメーターの型を判断します。

```
間違いの例:
char *jobscript_p = "./job.sh";
char *node_p = "8x8";
pjcmd_jobmodel_t jobmodel = PJCMD_JOBMODEL_BULK;

pjcmd_submit_put_param(handle_p, PJCMD_SUBMIT_SCRIPTFILE, jobscript_p); // 値(ポインター)を渡している
pjcmd_submit_put_job_resource(handle_p, "node","8x8");                  // 値(ポインター)を渡している
pjcmd_submit_put_param(handle_p, PJCMD_SUBMIT_JOBMODEL, jobmodel);      // 値(数値)を渡している
```

- 関数に渡されたパラメーターは、関数内でハンドルにコピーされます。このため、関数が成功すれば、元のパラメーターを変更してもハンドルの内容には影響ありません。

ジョブの削除のように、対象ジョブを指定する操作では、関数pjcmd_put_job()を使ってハンドルにジョブIDを設定します。

以下は、ジョブIDが10の通常ジョブをハンドルに設定する例です。

```
int64_t jobid[2] = {10, -1};  // ジョブIDを配列で指定。終端は -1

pjcmd_put_job(handle_p, jobid, 0, 0, NULL, 0, 0, PJCMD_JOBMODEL_NORMAL);
```

文字列で指定されたジョブIDからもハンドルにジョブIDを設定できます。

```
char *jobid_str_p = "10[1-5]";  // バルクジョブ 10[1]から10[5]

pjcmd_put_job_by_str(handle_p, jobid_str_p);
```

これを利用すれば、コマンドライン引数で指定されたジョブIDを数値に変換せずに、そのままハンドルに設定できます。

2.2.3 コマンドラインオプションの処理

コマンドAPIは、ジョブ運用管理機能のコマンドと同じオプションを認識し、ハンドルにパラメーターを設定する機能があります。

2.2.3.1 コマンドの引数からのパラメーター設定

"2.2.2 パラメーターの設定" では、利用者側で個々のパラメーターを設定していますが、ジョブ運用管理機能が提供するコマンドと同じオプション体系を採用するコマンドを作る場合は、コマンドの引数をコマンドAPI側でまとめて解析し、ハンドルに設定させることもできます。

例えば、作成するコマンドのオプションの仕様をpjsubコマンドと同じにする場合、関数pjcmd_submit_parse_pjsub_args()で引数を解析し、ハンドルにパラメーターを設定できます。

```
main(int argc, char **argv_pp)
{
    PjcmdHandle_t *handle_p;
    char *jobscript_p;
    ...
    pjcmd_submit_parse_pjsub_args(handle_p, argc, argv_pp);
    if (pjcmd_optind >= argc) {
        エラー処理;
```

```
    }
    jobscript_p = argv_pp[pjcmd_optind];
    pjcmd_submit_put_param(handle_p, PJCMD_SUBMIT_SCRIPTFILE, &jobscript_p);
    ...
```

関数pjcmd_submit_parse_pjsub_args()が正常終了すると、コマンドAPIのグローバル変数pjcmd_optindは引数の配列argv_pp[]におけるオプション以外の最初の引数を指します。オプション以外の引数については、呼び出し側で処理してください。
以下の例では、関数pjcmd_submit_parse_pjsub_args()でオプションをハンドルに設定したあと、残っている引数があれば、argv_pp[pjcmd_optind]をジョブスクリプトとみなし、ハンドルに設定しています(それ以降の引数は無視)。

```
    ...
    if (pjcmd_optind >= argc) { // 残っている引数がない
        ... // エラー処理
    }
    jobscript_p = argv_pp[pjcmd_optind];
    pjcmd_submit_put_param(handle_p, PJCMD_SUBMIT_SCRIPTFILE, &jobscript_p);  // ハンドルに
                                                                              // ジョブスクリプトを設定
    ...
```

2.2.3.2 コマンドラインパーサ

コマンドAPIが認識するオプションは、名前の変更や無効化ができます。これにより、作成するコマンドでジョブ運用管理機能のコマンドのオプションを継承しつつ、別のオプション名にしたり、不要なオプションを無効化したりできます。

コマンドAPIでは、オプションの仕様(オプション名、意味、および引数を持つか否か)はコマンドラインパーサと呼ぶ情報で扱います。オプション名の変更や無効化をする場合は、まずコマンドラインパーサを生成し、それに対して設定をします。

注意

- コマンドラインパーサはコマンドAPIを利用するプログラム内で有効な機能であり、ジョブ運用管理機能のコマンドのオプションを操作する機能ではありません。
- 現在、コマンドラインパーサはジョブ投入操作に関するpjsubコマンド相当のオプションだけサポートしています。

以下は、コマンドAPIが認識するpjsubコマンドのオプション-Lを-Rに変更し、オプション--fsと--appnameを無効化する例です。

```
PjcmdHandle_t *handle_p;
PjcmdParser_t *parser_p;

handle_p = pjcmd_create_handle(PJCMD_SUBMIT);              // ジョブ投入用のハンドル生成
parser_p = pjcmd_submit_create_pjsub_parser(handle_p);     // ジョブ投入に関するコマンドラインパーサ生成

pjcmd_renameopt_in_parser(parser_p, 'L', 'R', NULL, NULL); // オプション -L を -R に変更
pjcmd_delopt_in_parser(parser_p, 0, "fs");                 // オプション --fs を無効化
pjcmd_delopt_in_parser(parser_p, 0, "appname");            // オプション --appname を無効化

pjcmd_submit_parse_pjsub_args(handle_p, argc, argv_pp);    // 上記設定で、オプションを解析する
```

注意

- パーサはハンドルと結びついているため、オプション解析をする前に対応するハンドルを解放しないでください。ハンドルを先に解放した場合の動作は不定です。
- パーサによってオプション名を変更したり、無効化したりした場合、パーサに結びついているハンドルを指定する関数pjcmd_submit_parse_pjsub_args()での引数解析にも反映されます。

2.2.3.3 オプションの解析

ジョブ運用ソフトウェアが提供するコマンドのオプションに加えて、独自のオプションを持つコマンドを作成したい場合、関数pjcmd_getopt_long()を使うと引数の解析処理は独自オプションの解析だけで済みます。
関数pjcmd_getopt_long()はC言語ライブラリ関数getopt_long()に似ていますが、コマンドAPIが認識するコマンドのオプションを検出すると復帰せずに内部でそれを解析し、コマンドラインパーサに紐付けられているハンドルにパラメーターを設定します。
独自オプションのようにコマンドAPIが認識できないオプションを検出すると、関数pjcmd_getopt_long()は関数getopt_long()と同様の動作になり、復帰します。

以下は、ジョブを投入する処理で、ジョブ運用管理機能のpjsubコマンド相当のオプションから一部のオプションを変更または無効にし、独自の -Q オプションを処理する場合の例です。

```
main(int argc, char **argv_pp)
{
    int c, Q_flag = 0;
    char *myopts_p = "Q";  // 独自オプション -Q
    PjcmdHandle_t *handle_p;
    PjcmdParser_t *parser_p;
    ...

    handle_p = pjcmd_create_handle(PJCMD_SUBMIT);               // ハンドル生成
    parser_p = pjcmd_submit_create_pjsub_parser(handle_p);      // コマンドラインパーサ作成
    pjcmd_renameopt_in_parser(parser_p, 'L', 'R', NULL, NULL); // オプション -L を -R に変更
    pjcmd_delopt_in_parser(parser_p, 0, "fs");                  // オプション --fs を無効化
    pjcmd_delopt_in_parser(parser_p, 0, "appname");             // オプション --appname を無効化

    // コマンドライン引数 argv の解析とハンドルへの設定
    // 認識オプションは pjsubコマンドと同じオプション(parser_p) および -Q (myopts_p)
    while ((c = pjcmd_getopt_long(parser_p, argc, argv_pp, myopts_p, NULL, NULL) != -1) {
        // pjcmd_getopt_long() は、コマンドラインパーサが認識するオプションについては復帰しない。
        if (c == -1)
            break;
        switch (c) {
        case 'Q':  // 独自オプション -Q
            Q_flag = 1;
            break;
        default:   // コマンドラインパーサが認識するオプションでも、独自オプションでもない。
                   // 無効化したオプションは不明オプションとして扱われる。
            fprintf(stderr, "%c: Unknown option¥n", c);
            break;
    }
    ...
    if (Q_flag) {
        // -Q オプション指定時の処理
    }
    ...
}
```

注意

現在、コマンドラインパーサがサポートするのはpjsubコマンドのオプションだけです。したがって、コマンドラインパーサを必要とする関数pjcmd_getopt_long()で解析できるのもpjsubコマンドのオプションだけです。

2.2.4 ジョブ運用管理機能への依頼

2.2.4.1 操作依頼関数の呼び出し

ハンドルにパラメーターを設定したら、操作依頼関数でジョブ運用管理機能に対して操作依頼をします。

例えば、ジョブの投入の依頼は以下のようにします。

```
PjcmdHandle_t *handle_p
PjcmdResp_t *resp_p;
...
resp_p = pjcmd_submit_execute(handle_p);
```

操作結果である応答情報(PjcmdResp_t)が関数の復帰値で返されます。

注意

- ハンドルには必要なパラメーターをすべて設定した状態で、操作依頼をしてください。操作に必須のパラメーターが設定されていない場合は、エラーになります。
  例えば、バルクジョブを投入するときにバルク開始番号が設定されていない場合や、ジョブを削除するときにジョブIDが設定されていない場合です。
- 操作依頼関数を連続して呼び出した場合、前回の呼び出しから一定の時間が経過するまではジョブ運用管理機能への依頼は関数内で保留され、その間、関数は復帰しません。これはジョブ運用管理機能の負荷が上がるのを避けるためです。この時間の長さは管理者が変更できます。設定方法はマニュアル「ジョブ運用ソフトウェア 管理者向けガイド ジョブ管理編」の"第3章 ジョブ運用管理機能の設定"にある"コマンドAPIの設定"を参照してください。
- 操作依頼ごとにコマンドAPIはジョブ運用管理機能と通信します。このため、1つのプログラムで、ジョブ運用管理機能への操作依頼が完了しない間は新たな操作依頼は受け付けられません。
  - ー すでにほかのスレッドで操作依頼関数を処理しているときは、新たな操作依頼関数の呼び出しはエラーになり、pjcmd_errcodeにPJCMD_ERROR_BUSYが設定されます。処理中の操作依頼関数が終了してから、再度呼び出してください。
  - ー 操作依頼関数が復帰しても、応答情報を解放するまで、コマンドAPIは操作内容に関してジョブ運用管理機能と内部で通信をしている場合があります。このため、操作要求関数を呼び出したあとで、ほかの操作依頼関数を呼び出す場合は、先に応答情報を解放してください。
- 操作依頼関数内では、一部のシグナルに対する動作は以下のようになります。

```
SIGHUP、SIGQUIT、SIGINT、 SIGALRM、SIGTERM : 処理中断
SIGUSR1、SIGUSR2、SIGPOLL、SIGPROF、SIGVTALRM、SIGIO、SIGPWR: 無視
```

  シグナルによって操作依頼関数の処理が中断した場合、エラーコードpjcmd_errcodeがPJCMD_ERR_SIGNALになります。
  関数復帰後に、シグナルの設定は呼び出し前の内容に戻されます。

2.2.4.2 ジョブの投入について

ジョブの投入は、ジョブモデルによってはほかの操作より特殊な手順を踏む場合があります。以下では、コマンドAPIで投入する場合の注意事項を説明します。

- ステップジョブ
  ステップジョブのサブジョブを投入する場合、サブジョブごとに1つのハンドルを用意し、投入操作をしてください。
  同じステップジョブの場合は、先行するサブジョブのハンドルを複製して、必要なパラメーターだけ更新すればよいです。

  また、1つのステップジョブの複数のサブジョブをまとめて投入する場合は、関数pjcmd_submit_executev()を利用できます。
  例えばpjsubコマンドでステップジョブの複数のサブジョブをまとめて投入する場合、以下のように実行します。

```
$ pjsub --step -N myjob stepjob0.sh stepjob1.sh stepjob2.sh stepjob3.sh stepjob4.sh
```

  これと同様なことをコマンドAPIで行う場合は、以下のようにできます。

```
#define SUBJOB_NUM 5

PjcmdHandle_t *handle_p[SUBJOB_NUM];
PjcmdResp_t *resp_p;
int i;
pjcmd_jobmodel_t model = PJCMD_JOBMODEL_STEP;
char *jobname_p = "myjob";
char *jobscript_p[SUBJOB_NUM] = {"stepjob0.sh", "stepjob1.sh", "stepjob2.sh",  "stepjob3.sh", "stepjob4.sh"};

handle_p[0] = pjcmd_create_handle(PJCMD_SUBMIT);
```

```
pjcmd_submit_put_param(handle_p[0], PJCMD_SUBMIT_JOBMODEL, &model);
pjcmd_submit_put_param(handle_p[0], PJCMD_SUBMIT_JOBNAME, &jobname_p);
pjcmd_submit_put_param(handle_p[0], PJCMD_SUBMIT_SCRIPTFILE, jobscript_p[0]);
...

for (i = 1; i < subjob_num; i++) {
    handle_p[i] = pjcmd_clone_handle(&handle_p[0]); // 最初のサブジョブのハンドルを複製
    pjcmd_submit_put_param(handle_p[i], PJCMD_SUBMIT_SCRIPTFILE, jobscript_p[i]); // ジョブスクリプト
}

resp_p = pjcmd_submit_executev(handle_p, SUBJOB_NUM); // サブジョブのハンドルをまとめて指定し、投入
```

- 会話型ジョブ

  会話型ジョブの場合、関数pjcmd_submit_execute() で投入依頼をすると、会話型ジョブの入力がクローズされる(会話型ジョブが終了する)まで関数は復帰しません。

  また、会話型ジョブの投入は、内部で以下の4つの段階があります。

  1. 会話型ジョブの受付が完了する。
  2. 会話型ジョブが実行待ちの状態になる。
  3. 会話型ジョブの実行が開始される。
  4. 会話型ジョブが終了する。

  ジョブ運用管理機能のpjsubコマンドは、これらの各段階で処理の経過を示すメッセージを出力しています。コマンドAPIでは、各段階で利用者が用意した関数を呼び出すコールバックの仕組みを用意しています。コマンドAPIを利用したコマンドでpjsubコマンドと同様なメッセージ出力処理をしたい場合は、関数pjcmd_submit_set_callback()でメッセージを出力する関数を登録してから、ジョブ投入の操作依頼をしてください。コールバック関数の引数にはサブジョブID構造体(PjcmdSubjobid_t)が渡されます。

  ここでは、pjsubコマンドで会話型ジョブを投入したときに表示される以下のメッセージと同様な出力をするコールバック関数の例を挙げます(エラー処理などは省略しています)。

```
$ pjsub --interact
[INFO] PJM 0000 pjsub Job 405916 submitted.              ← 会話型ジョブ投入を示すメッセージ
[INFO] PJM 0081 .connected.                              ← 会話型ジョブの準備中を示すメッセージ
[INFO] PJM 0082 pjsub Interactive job 405916 started.    ← 会話型ジョブ開始を示すメッセージ
$ ← 会話型ジョブ内のシェルプロンプト
...
$ exit ← シェルの終了
[INFO] PJM 0083 pjsub Interactive job 405916 completed. ← 会話型ジョブ終了を示すメッセージ
```

  a. 会話型ジョブ投入を示すメッセージ
     会話型ジョブの受付が完了したときに呼ばれるコールバック関数で表示します。コールバック関数の引数として渡されるサブジョブID構造体(PjcmdSubjobid_t)から文字列のジョブIDを作成し(上記例では "405916")、表示します。

```
void interact_job_accept_msg(const PjcmdSubjobid_t *subjobid_p)
{
    char buf[PJCMD_MAX_SUBJOBID_STR_LEN];

    pjcmd_subjobid_to_str(subjobid_p, buf);
    fprintf(stdout, "[INFO] PJM 0000 mypjsub Job %s submitted.¥n", buf);
}
```

  b. 会話型ジョブの準備中を示すメッセージ
     会話型ジョブが実行待ちの間、定期的(3秒間隔)に呼ばれるコールバック関数で表示します。この例では、呼ばれるたびに "."を表示します。

```
int interact_job_wait_msg_called = 0;
void interact_job_wait_msg(const PjcmdSubjobid_t *subjobid_p)
{
    if (interact_job_wait_msg_called != 0) {
        fprintf(stdout, "[INFO] PJM 0081 "); // 初回呼び出し時はメッセージ先頭だけを表示
        interact_job_wait_msg_called = 1;
```

```
    }
    fprintf(stdout, ".");
    fflush(stdout);
}
```

c. 会話型ジョブ開始を示すメッセージ
会話型ジョブが開始されるときに呼ばれるコールバック関数で表示します。コールバック関数の引数として渡されるサブジョブID構造体(PjcmdSubjobid_t)から文字列のジョブIDを作成し(上記例では "405916")、表示します。

```
void interact_job_start_msg(const PjcmdSubjobid_t *subjobid_p)
{
    char buf[PJCMD_MAX_SUBJOBID_STR_LEN];

    fprintf(stdout, "connected¥n"); // ジョブの準備中メッセージの終りとして表示
    pjcmd_subjobid_to_str(subjobid_p, buf);
    fprintf(stdout, "[INFO] PJM 0082 mypjsub Interactive job %s started.¥n", buf);
}
```

d. 会話型ジョブ終了を示すメッセージ
会話型ジョブが終了したときに呼ばれるコールバック関数で表示します。コールバック関数の引数として渡されるサブジョブID構造体(PjcmdSubjobid_t)から文字列のジョブIDを作成し(上記例では "405916")、表示します。

```
void interact_job_end_msg(const PjcmdSubjobid_t *subjobid_p)
{
    char buf[PJCMD_MAX_SUBJOBID_STR_LEN];

    pjcmd_subjobid_to_str(subjobid_p, buf);
    fprintf(stdout, "[INFO] PJM 0082 mypjsub Interactive job %s completed.¥n", buf);
}
```

上記のように用意したコールバック関数を会話型ジョブの投入を依頼する前に、以下のように登録しておきます。

```
if (jobtype == PJCMD_JOBTYPE_INTERACTIVE) {              // 会話型ジョブの場合
    pjcmd_submit_set_callback(handle_p,                  // ハンドル
                              &interact_job_accept_msg,  // ジョブ受付時
                              &interact_job_wait_msg,    // ジョブ開始待ち時
                              &interact_job_start_msg,   // ジョブ開始時
                              &interact_job_end_msg);    // ジョブ終了時
}
...
resp_p = pjcmd_submit_execute(handle_p);
...
```

これにより、依頼関数pjcmd_submit_execute()内の処理の段階ごとに、コールバック関数が呼び出されます。

- 標準入力からのジョブ

pjsubコマンドは、ジョブスクリプトを指定せずに、標準入力からジョブの内容を与えることができます。コマンドAPIで同様なことをする場合は、プログラム内で標準入力の内容を一時的なファイルに格納し、ジョブスクリプトとしてハンドルに設定してください。

なお、コマンドAPIには標準入力の内容を一時的なジョブスクリプトとして作成するための関数pjcmd_submit_create_scriptfile_from_stdin()があります。この関数を利用した例は以下のようになります。

```
char *tmp_jobscript_p;

tmp_jobscript_p = pjcmd_submit_create_scriptfile_from_stdin(NULL, NULL); // 自動生成したファイルに入力を格納
if (tmp_jobscript_p == NULL) {
    // エラー処理へ
}
pjcmd_submit_put_param(handle_p, PJCMD_SUBMIT_SCRIPTFILE, &tmp_jobscript_p); // ジョブスクリプト名を設定
...
pjcmd_submit_execute(handle_p);
...
unlink(tmp_jobscipt_p); // 一時的に作成したジョブスクリプトを削除
```

2.2.4.3 ジョブの操作について

以下のジョブ操作では、ジョブ運用管理機能への処理の依頼が受け付けられ、応答が返るまでに時間がかかる場合があります。

- ジョブの削除
- ジョブの固定
- ジョブの固定解除
- ジョブへのシグナル送信
- ジョブのパラメーター変更

上記のジョブ操作関数に該当するジョブ運用管理機能のジョブ操作コマンドでは、依頼が受け付けられるまでに時間がかかる場合、処理の経過を示すメッセージを出力しています。コマンドAPIでは、依頼が受け付けられるまでに時間がかかる場合、利用者が用意した関数を呼び出すコールバックの仕組みを用意しています。

コマンドAPIを利用したコマンドで各種操作コマンドと同様なメッセージ出力処理をしたい場合は、関数pjcmd_*xxxx*_set_callback()でメッセージを出力する関数を登録してから、ジョブの操作依頼をしてください。

ここでは、pjdelコマンドを実行したときに表示される以下のメッセージと同様な出力をするコールバック関数の例を挙げます(エラー処理などは省略しています)。

```
$ pjdel 405916
[INFO] PJM 0181 ...done.        ←"[INFO] PJM 0181 ..." : ジョブの削除依頼の受付け待ちを示すメッセージ
                                  "done."               : ジョブの削除依頼の受付けが完了したことを示すメッセージ
[INFO] PJM 0100 pjdel Accepted job 405916.
```

a. ジョブの削除依頼の受付け待ちを示すメッセージ

ジョブの削除依頼の受付けを待つ間、定期的(3秒間隔)に呼ばれるコールバック関数で表示します。この例では、呼ばれるたびに"."を表示します。

```
int kill_wait_msg_called = 0;
void kill_wait_msg(void)
{
    if (kill_wait_msg_called == 0) {
        fprintf(stdout, "[INFO] PJM 0181 "); // 初回呼び出し時はメッセージ先頭を表示
        kill_wait_msg_called = 1;
    }
    fprintf(stdout, ".");
    fflush(stdout);
}
```

b. ジョブの削除依頼の受付けが完了したことを示すメッセージ

ジョブの削除依頼の受付けが完了したときに呼ばれるコールバック関数で表示します。

```
void kill_accept_msg(void)
{
    fprintf(stdout, "done.¥n"); // ジョブの削除依頼の応答待ちメッセージの終りとして表示
}
```

上記のように用意したコールバック関数をジョブの削除依頼をする前に、以下のように登録しておきます。

```
pjcmd_kill_set_callback(handle_p,           // ハンドル
                        &kill_wait_msg,     // ジョブの削除依頼の受付け待ち時
                        &kill_accept_msg); // ジョブの削除依頼の受付け待ちの完了時
}
...
resp_p = pjcmd_kill_execute(handle_p);
```

これにより、依頼関数pjcmd_kill_execute()内の処理の段階ごとに、コールバック関数が呼び出されます。

2.2.5 結果の参照

2.2.5.1 応答情報の参照

操作依頼の結果は、応答情報を参照することでわかります。

```
PjcmdHandle_t *handle_p;
PjcmdResp_t *resp_p;
int code, subcode;
char *detail_p;
...
resp_p = pjcmd_submit_execute(handle_p);                    // ジョブ投入操作要求
if (resp_p == NULL) {
    // エラー処理へ
}

pjcmd_get_result(resp_p, &code, &subcode, &detail_p);  // 操作依頼の結果を取得
if (code != 0) {
    //エラー処理へ
}
...
```

a. 応答情報がNULLの場合
操作依頼関数の復帰値として返される応答情報がNULLの場合は、操作依頼をする前のチェック、例えばハンドルの内容に不備があった場合に発生します。

b. 応答情報がNULLでない場合
応答情報が返された場合、まず、関数pjcmd_get_result()で操作依頼が成功したかどうかを確認します。第2引数(code)で0(成功)が返された場合、さらに以降で説明する詳細な結果を参照できます。

応答情報から得られる詳細な情報は、操作内容がジョブ操作とそれ以外とで異なります。

・ ジョブ操作

ジョブの投入やジョブの削除のようなジョブ操作では、応答情報から以下の情報が得られます。

ー 操作したジョブ数

関数pjcmd_get_jobresult_num()で、操作対象の総ジョブ数と操作が成功したジョブ数がわかります。

```
PjcmdHandle_t *handle_p;
PjcmdResp_t *resp_p;
int64_t num[2]; // 総ジョブ数と成功したジョブ数を格納する配列
...
resp_p = pjcmd_submit_execute(handle_p);                    // ジョブ投入操作要求
if (resp_p == NULL) {
    // エラー処理へ
}
pjcmd_get_result(resp_p, &code, &subcode, &detail_p);  // 操作依頼の結果を取得
if (code != 0) {
    // エラー処理へ
}
pjcmd_get_jobresult_num(resp_p, num);                       // 投入が成功したジョブ数を取得
```

上記の場合、投入した総ジョブ数がnum[0]に、投入が成功したジョブ数がnum[1]に格納されます。

ー 個々のジョブの結果

関数pjcmd_get_jobresult_info()で、個々のジョブの操作結果がわかります。

```
int code, subcode;
char *detail_p;
PjcmdSubjobid_t *subjobid_p;
...
resp_p = pjcmd_submit_execute(handle_p);                    // ジョブ投入操作要求
```

```
if (resp == NULL) {
    // エラー処理へ
}
pjcmd_get_result(resp_p, &code, &subcode, &detail_p);  // 操作依頼の結果を取得
if (code != 0) {
    // エラー処理へ
}
pjcmd_get_jobresult_num(resp_p, num);                  // 投入した総ジョブ数と
                                                       // 投入が成功したジョブ数を取得

for (i = 0; i < num[0]; i++) {
    pjcmd_get_jobresult_info(resp_p, PJCMD_JOBRESULT_ANY, i, PJCMD_JOBRESULT_CODE, &code);
    pjcmd_get_jobresult_info(resp_p, PJCMD_JOBRESULT_ANY, i, PJCMD_JOBRESULT_SUB_CODE, &subcode);
    pjcmd_get_jobresult_info(resp_p, PJCMD_JOBRESULT_ANY, i, PJCMD_JOBRESULT_DETAIL, &detail_p);
    pjcmd_get_jobresult_info(resp_p, PJCMD_JOBRESULT_ANY, i, PJCMD_JOBRESULT_SUBJOBID, &subjobid_p);

    printf("....<取得した情報の表示など> ...");
}
```

上記では、関数pjcmd_get_jobresult_info()で、応答情報から操作対象のすべてのジョブについて順に、結果コード code、詳細結果コード subcode、詳細結果(文字列) detail、およびサブジョブID subjobidを取得しています。第2引数で、投入が成功したジョブだけ(PJCMD_JOBRESULT_OK)、または失敗したジョブだけ(PJCMD_JOBRESULT_ERR)を指定できます。

参考

関数 pjcmd_get_jobresult_info() では、結果を参照するジョブのインデックスを指定するため、事前に関数 pjcmd_get_jobresult_num()でジョブの数を取得してください。対象ジョブ数を超えるインデックスを指定するとエラーになります。

- ジョブ操作以外

ジョブ操作以外(情報取得など)の場合は、応答情報から操作固有の情報を参照できます。参照の仕方は情報の種類によって異なりますが、以下ではジョブ情報(pjstatコマンド相当)を参照する例を示します。

```
pjcmd_result_t ret;
PjcmdResp_t *resp_p;

...
resp_p = pjcmd_jobinfo_execute(handle_p);              // ジョブ投入操作要求
if (resp_p == NULL) {
    // エラー処理へ
}
pjcmd_get_result(resp_p, &code, &subcode, &detail_p);  // 操作依頼の結果を取得
if (code != 0) {
    // エラー処理へ
}
do (ret = pjcmd_jobinfo_read_infogrp(resp_p)) {        // 1つの情報グループへ移動
    if (ret == PJCMD_ERR) {
        if (pjcmd_errcode == PJCMD_ERROR_NODATA) {
            break;
        }
        fprintf(stderr, "%s: Cannot read infogrp¥n", CMD_NAME);
        pjcmd_destroy_resp(resp_p);
        pjcmd_destroy_handle(handle_p);
        exit(EXIT_FAILURE);
    }

    pjcmd_jobinfo_print_resp(resp_p, PJCMD_JOBINFO_PRINT_JOBINFO); // 情報グループの内容を表示
}
```

応答情報には、情報グループが複数格納されています。情報グループとは情報の取得単位で、pjstatコマンドの--ru、-rg オプションによるリソースユニット、リソースグループごとの表示に相当する情報です。応答情報の中には、現在参照している情報グループを指

すポインターがあり、関数pjcmd_jobinfo_read_infogrp()は呼ばれるたびに、このポインターを次の情報グループに更新します。
次に、関数pjcmd_jobinfo_print_resp()を使って、現情報グループのサマリ情報とジョブ情報を表示させています。

参考

ー 上記は、pjstat コマンドを以下のように実行する場合の表示に相当します。

```
$ pjstat --ru --rg
[ RSCUNIT: unit1 ]          ※リソースユニット、リソースグループごとの表示単位が情報グループに相当
[ RSCGRP: group1 ]
JOB_ID     JOB_NAME   ...
2927       jobA       ...

[ RSCUNIT: unit1 ]
[ RSCGRP: group2 ]
JOB_ID     JOB_NAME   ...
2928       jobB       ...
...
```

ー 1つずつジョブの情報を確認したい場合は、関数pjcmd_jobinfo_read_jobinfo()を使います。この関数を使うと、現在の情報グループに属するジョブの情報を1つずつ参照できます。

2.2.5.2 エラー情報

コマンドAPIのエラーを検出するには以下の手段があります。

- 関数の復帰値 (pjcmd_result_t型やポインター型)
  関数の復帰値によって、関数が成功したか、失敗したかがわかります。
- エラーコード pjcmd_errcode
  関数が失敗した場合、その内容を示すコードが設定されます。
- 応答情報
  操作依頼の結果は、応答情報の内容を参照することでわかります("2.2.5.1 応答情報の参照"参照)。
- 詳細エラー情報
  上記の情報よりもさらに詳細にエラーの内容がわかります。

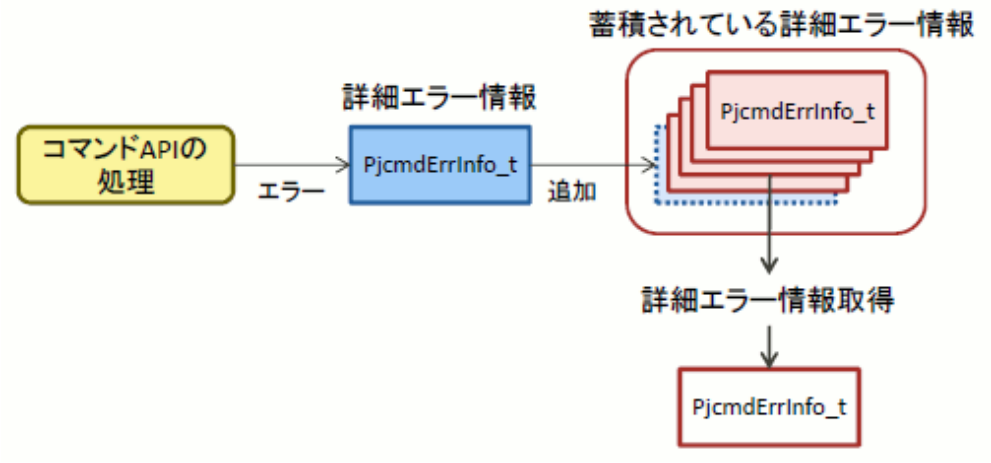
参考

標準コマンドのメッセージに含まれるような情報が必要な場合、関数復帰値やエラーコードだけでは不十分で、詳細エラー情報を参照する必要があります。

以下では、詳細エラー情報について説明します。

詳細エラー情報(PjcmdErrInfo_t)は、コマンドAPI内でエラーが発生すると蓄積されていきます。エラーによっては、1度に複数の詳細エラー情報が蓄積することもあります。

図2.1 詳細エラー情報のイメージ



1つの詳細エラー情報には、以下の情報が格納されます。

表2.1 詳細エラー情報に含まれる情報

情報	説明
詳細エラーコード	エラーの種類(pjcmd_suberrcode_t)を示します。
詳細コード1～3	エラーの詳細コード(数値)。エラーの種類によって最大3つまで設定されます。
詳細情報(文字列)1～5	エラーの詳細情報(文字列)。エラーの種類によって最大5つまで設定されます。
エラーが発生したジョブスクリプトの行番号	エラーがジョブスクリプト内で発生した場合に、その行番号を示します。
サブジョブID	特定のジョブに関するエラーの場合は、そのジョブのジョブID、バルク番号、ステップ番号を示します。

参照

詳細エラーコードごとに、取得できる詳細コードと詳細情報(文字列)は異なります。詳細は"付録Aリファレンス:コマンドAPI共通" の "A.7.2 詳細エラーコード"を参照してください。

コマンドAPIの利用側は、関数のエラーを検出すると、詳細エラー情報を参照し、必要があればメッセージに加工して表示します。

詳細エラー情報を参照する関数には以下があります。

表2.2 詳細エラー情報を参照する関数

関数	説明
pjcmd_error_read_errinfo()	コマンドAPI内に蓄積されている1つの詳細エラー情報を取り出します。 呼び出すたびに、次の詳細エラー情報を返します。
pjcmd_error_read_errinfo_by_sjid()	コマンドAPI内に蓄積されている詳細エラー情報がジョブに対応する内容である場合、指定したサブジョブID構造体に対応する詳細エラー情報を返します。
pjcmd_error_get_info()	詳細エラー情報内の情報(詳細情報(文字列)以外)を参照します。
pjcmd_error_get_detail_info()	詳細エラー情報内の詳細情報(文字列)を参照します。
pjcmd_error_destroy_errinfo()	1つの詳細エラー情報を解放(削除)します。
pjcmd_error_clear_errinfo()	蓄積されているすべての詳細エラー情報を解放(削除)します。

注意

詳細エラー情報は、プロセスが使用するメモリ内に蓄積され続けます。このため、通常はエラー発生時に詳細エラー情報を参照し、エラー情報の出力などをした後、解放(削除)してください。
多くの場合、エラーが発生するとプログラムは終了するように作られますが、このような場合、詳細エラー情報は明に解放しなくても、プログラム終了時にOSによって解放されます。

[詳細エラー情報の参照例]

コマンドAPIの関数がエラーになったときに、詳細エラー情報を参照するコードの例を以下に示します。

```
 1  main(int argc, char **argv)
 2  {
 3      PjcmdHandle_p *handle_p;
 4      PjcmdResp_p *resp_p;
 5      int code, subcode;
 6      char *detail_p;
 7      ...
 8      ret = pjcmd_submit_parse_pjsub_args(handle_p, argc, argv);
 9      if (ret != PJCMD_OK) { // パラメーター設定がエラー
10          goto l_err;
11      }
12      ...
13      resp_p = pjcmd_submit_execute(handle_p);
14      if (resp_p == NULL) { // 投入依頼失敗
15          goto l_err;
16      }
17
18      if (pjcmd_get_result(resp_p, &code, &subcode, &detail_p) != PJCMD_OK) {
19          goto l_err;  // 結果の取得失敗
20      }
21
22      if (code != 0) { // 依頼した操作が正常に終了しなかった
23          goto l_err;
24      }
25
26  ...
27
28  l_err:
29      mycmd_print_error();  // エラーメッセージ表示
30      pjcmd_error_destroy_errinfo();
31      pjcmd_destroy_resp(resp_p);
32      pjcmd_destroy_handle(handle_p);
33      ...
34      exit(1);
35  }
36
37  void mycmd_print_error(void)
38  {
39      PjcmdErrInfo_t *einfo_p;
40      int suberrcode, code1, code2, code3, line;
41      char *detail_p[5];
42      PjcmdSubjobid_t *subjobid_p;
43
44      while ((errinfo_p = pjcmd_error_read_errinfo()) != NULL) {
45          pjcmd_error_get_info(einfo_p, PJCMD_ERRINFO_SUBERRCODE, &suberrcode);
46          switch (suberrcode) {
47          case PJCMD_SUBERR_UNKNOWN_OPT: // 不明なオプションを検出
48              fprintf(stderr, "[ERR.] 0001 mycmd Unknown option '%s'¥n",
49                              pjcmd_error_get_detail_info(einfo_p, 0)); // 不明なオプション
50              break;
```

```
51          case PJCMD_SUBERR_COMBINATION: // オプションの組み合わせが不正
52              detail_p[0] = pjcmd_error_get_detail_info(einfo_p, 0); // オプション1
53              detail_p[1] = pjcmd_error_get_detail_info(einfo_p, 1); // オプション2
54              fprintf(stderr, "[ERR.] 0002 mycmd Invalid combination of options: '%s' and '%s'¥n",
55                              detail_p[0], detail_p[1]);
56              break;
57          ...
58          case PJCMD_SUBERR_UPPER_LIMIT: // 指定した資源量が上限を超えている
59              pjcmd_error_get_info(einfo_p, PJCMD_ERRINFO_SJID, &subjobid_p);
60              detail_p[0] = pjcmd_error_get_detail_info(einfo_p, 0); // 資源名
61              detail_p[1] = pjcmd_error_get_detail_info(einfo_p, 1); // 指定した資源量
62              detail_p[2] = pjcmd_error_get_detail_info(einfo_p, 2); // 上限値
63              fprintf(stderr, "[ERR.] 0057 mycmd %s=%s is greater than the upper limit (%s).¥n",
64                              detail_p[0], detail_p[1], detail_p[2]);
65              break;
66          ...
67          case PJCMD_SUBERR_DAEMON_ISNOT_PRESENT: // ジョブマネージャー機能が
68                                                  // 動作していない/通信ができない
69              fprintf(stderr, "[ERR.] 0090 mycmd Job manager does not work¥n");
70              break;
71          ...
72          }
73      }
74  }
```

- 10、15、19、および23行目
  コマンドAPIの関数でエラーが発生すると、エラー処理にジャンプします。
- 29行目
  詳細エラー情報によるエラーメッセージ表示関数を呼び出します。
- 30～34行目
  エラーメッセージ表示後、詳細エラー情報、応答情報、およびハンドルを解放し、終了します。
- 44～73行目
  詳細エラー情報PjcmdErrInfo_tを1つずつ読みます。
- 45行目
  詳細エラー情報から、詳細エラーコードを取得します。
- 46～72行目
  すべての詳細エラーコードについて、付随する情報を取得し、メッセージとして表示します。
  詳細エラーコードの種類と付随する情報の種類は"付録A リファレンス:コマンドAPI共通" の "A.7.2 詳細エラーコード"を参照してください。

  例えば、58～65行目は、指定した資源の量がジョブACL機能で定義されている上限を超えている場合のエラー処理です。関数pjcmd_error_get_detail_info()で、資源名、指定した資源量、およびその上限値を詳細エラー情報から取得し、関数fprintf()で表示しています。

2.2.6 ハンドル、コマンドラインパーサ、および応答情報の解放

処理が終わり、プログラムが終了するときは、生成したハンドル、コマンドラインパーサ、および応答情報を解放してください。

```
PjcmdHandle_t *handle_p;
PjcmdResp_t *resp_p;
PjcmdParser_t *parser_p;
...
parser_p = pjcmd_submit_create_pjsub_parser(handle_p);
...
resp_p = pjcmd_submit_execute(handle_p);
...
pjcmd_submit_destroy_pjsub_parser(parser_p);
```

```
pjcmd_destroy_resp(resp_p);
pjcmd_destroy_handle(handle_p);
```

注意

ハンドル、応答情報、およびコマンドラインパーサを解放する場合、ハンドルは最後に解放してください。応答情報とコマンドラインパーサにはハンドルが紐付けされているため、先にハンドルを解放したあとで応答情報やコマンドラインパーサを操作すると動作は不定です。
なお、これらの領域はプログラムのデータ領域の一部ですので、解放せずにプログラムが終了した場合、OSによって領域は解放されます。

2.2.7 コマンドAPI利用時の注意

- コマンドAPIはマルチスレッドのプログラムで利用できます。ただし、1つのハンドルを複数のスレッドで同時に更新しないでください。

2.3 コマンドの作成

コマンドAPIのライブラリは、ログインノード、計算クラスタ管理ノード、およびシステム管理ノードの以下にあります。

```
/usr/lib64/libpjcmd.so
```

作成したソースファイルをコンパイルして実行可能ファイルを作成するときは、コマンドAPIのライブラリlibpjcmdをリンクしてください。

```
$ gcc -o mycmd mycmd.c -lpjcmd ...
```

注意

コンパイラーは、OSの標準のgccを使用してください。それ以外のコンパイラーについてはサポートしていません。

2.4 コマンドAPIの設定

管理者は、コマンドAPIの動作を変更するための設定ができます。
デフォルトの設定値を変更する場合は、コマンドAPIが利用できるログインノード、計算クラスタ管理ノード、およびシステム管理ノードのpmpjcmd.confファイルを編集してください。

```
/etc/opt/FJSVtcs/pjm/pmpjcmd.conf
```

設定ファイル pmpjcmd.conf の内容は、コマンドAPIの関数を利用しているコマンドを実行するたびに最新の内容が読み込まれます。コマンドAPIは呼び出されたノード上の設定ファイルを読み込むため、各ノードを同じ設定値にしたい場合は、それぞれのノードの設定ファイルを同じ内容にしてください。

pmpjcmd.confファイルのパーミッションは以下のように設定してください。

- 所有者/グループ: root/root
- ファイルモード: 0644

参照

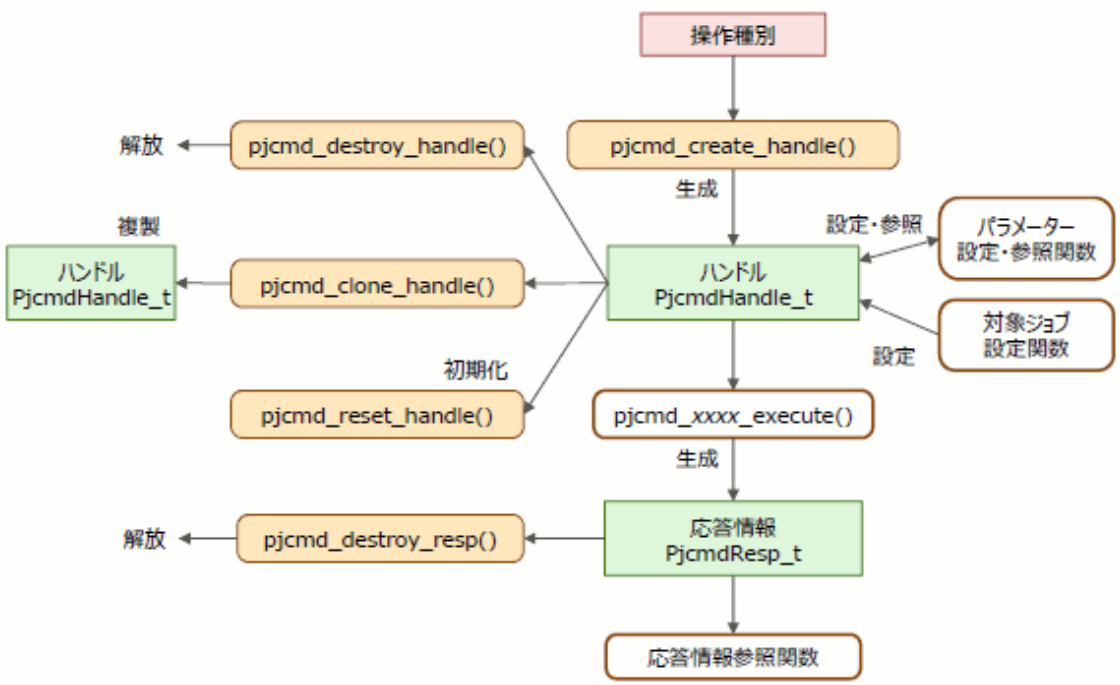
設定ファイルの詳細については、マニュアル「ジョブ運用ソフトウェア 管理者向けガイド ジョブ管理編」の"第3章 ジョブ運用管理機能の設定"にある"コマンドAPIの設定"を参照してください。

付録A　リファレンス:コマンドAPI共通

A.1　ハンドル、応答情報の操作

ここでは、ハンドルと応答情報を操作するための関数を説明します。

図A.1 ハンドルや応答情報の操作



A.1.1　pjcmd_create_handle()

```
PjcmdHandle_t *pjcmd_create_handle(pjcmd_operation_t op)
```

ハンドルを生成します。

[引数]

op

操作の種類を示す識別子

識別子	説明
PJCMD_SUBMIT	ジョブ投入
PJCMD_KILL	ジョブ削除
PJCMD_SIGNAL	シグナル送信
PJCMD_HOLD	ジョブ固定
PJCMD_RELEASE	ジョブ固定解除
PJCMD_ALTER	ジョブパラメーター変更
PJCMD_WAIT	ジョブ終了待ち合わせ
PJCMD_JOBINFO	ジョブ情報取得
PJCMD_RSCINFO	リソースユニット・リソースグループ資源情報取得
PJCMD_LIMITINFO	制限値情報取得

識別子	説明
PJCMD_RSCSTAT	資源状態取得
PJCMD_JOBACL	ジョブACL情報取得
PJCMD_SETPJMSTAT	ジョブ運用の変更(ジョブの投入・実行の許可の設定)
PJCMD_GETPJMSTAT	ジョブ運用の設定状況の取得(ジョブ投入・実行の許可状況の取得)

[返り値]

生成したハンドルへのポインターを返します。失敗した場合はNULLを返し、原因がpjcmd_errcodeに設定されます。生成されたハンドルは、呼び出し側が関数pjcmd_destroy_handle()を使用して解放してください。

[pjcmd_errcode]

PJCMD_ERROR_UNKNOWN_PARAM

*op*に不明な値が指定されました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

A.1.2 pjcmd_clone_handle()

```
PjcmdHandle_t *pjcmd_clone_handle(const PjcmdHandle_t *handle_p)
```

ハンドル *handle_p* を複製します。ジョブを投入する際に、別のジョブのパラメーターを流用して投入するような場合に利用します。

[引数]

handle_p

複製するハンドルへのポインター

[返り値]

複製したハンドルへのポインターを返します。失敗した場合はNULLを返し、原因がpjcmd_errcodeに設定されます。複製したハンドルは、呼び出し側が関数pjcmd_destroy_handle()を使用して解放してください。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

*handle_p*が不正(NULL)です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

A.1.3 pjcmd_reset_handle()

```
pjcmd_result_t pjcmd_reset_handle(PjcmdHandle_t *handle_p)
```

ハンドルの内容を初期化します。ハンドルは関数pjcmd_create_handle()で生成した初期状態と同じ内容になります。

[引数]

handle_p

初期化するハンドルへのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

*handle_p*が不正(NULL)です。

A.1.4 pjcmd_destroy_handle()

```
pjcmd_result_t pjcmd_destroy_handle(PjcmdHandle_t *handle_p)
```

ハンドルを解放します。

[引数]

handle_p

解放するハンドルへのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

handle_p が不正(NULL)です。

A.1.5 pjcmd_destroy_resp()

```
pjcmd_result_t pjcmd_destroy_resp(PjcmdResp_t *resp_p)
```

応答情報を解放します。応答情報は関数pjcmd_*operation*_execute()が返す結果情報です。

[引数]

resp_p

解放する応答情報へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

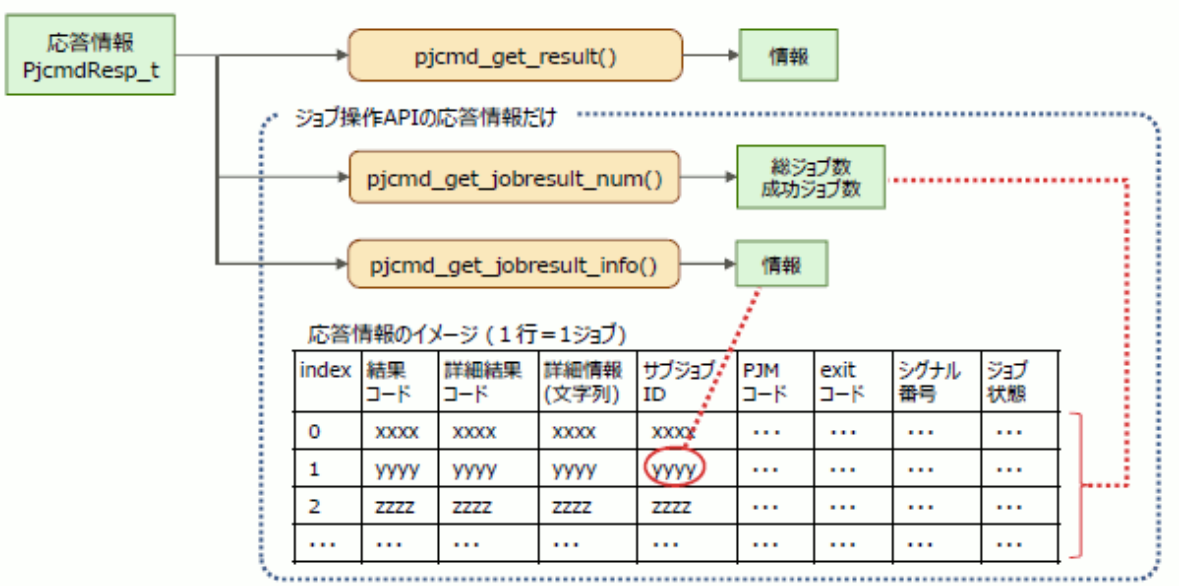
PJCMD_ERROR_INVALID_RESP

*resp_p*が不正(NULL)です。

A.2 操作結果の参照

ここでは、ジョブ操作や情報取得の応答情報から、結果についての情報を参照するための関数を説明します。

図A.2 操作や情報取得の結果の参照



A.2.1 pjcmd_get_result()

```
pjcmd_result_t pjcmd_get_result(const PjcmdResp_t *resp_p, int *code_p, int *subcode_p, char **detail_pp);
```

応答情報から、操作の結果コードや詳細情報を取得します。
この関数で得られる情報は操作や情報取得要求についての結果です。ジョブ操作(ジョブ投入、ジョブ削除、ジョブ固定、ジョブ固定解除、シグナル送信、ジョブ終了待ち合わせ、およびジョブパラメーター変更)における個々のジョブの結果については、関数pjcmd_get_jobresult_info()を利用してください。

[引数]

resp_p

応答情報へのポインター

code_p

code_p*に結果コードを格納します。code_p*が0の場合は操作が正常終了したことを示します。結果コードが0でない場合は、操作が正常に終了しなかったことを示し、(int)**subcode_p* および (char *) **detail_pp*には調査用の詳細情報が格納されます。

subcode_p

**subcode_p*に調査用の詳細結果コードを格納します。

detail_pp

**detail_pp*に調査用の詳細情報(文字列)へのポインターを格納します。

[返り値]

PJCMD_OK

結果情報の取得が成功。

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

*resp_p*が不正(NULL)です。

PJCMD_ERROR_INVALID_ARGUMENT

code_p、*subcode_p*、または*detail_pp*が不正(NULL)です。

A.2.2 pjcmd_get_jobresult_num()

```
pjcmd_result_t pjcmd_get_jobresult_num(const PjcmdResp_t *resp_p, int64_t *num_p)
```

ジョブ操作(*)の応答情報から、対象となった総ジョブ数と操作が成功したジョブ数を取得します。
(*)ジョブの投入、削除、固定、固定解除、シグナル送信、パラメーター変更、および終了待ち合わせ("付録B リファレンス:ジョブ操作API"参照)

[引数]

resp_p

応答情報へのポインター

num_p

配列*num_p*[]に総ジョブ数と操作が成功したジョブ数が格納されます。
num_p[0]: 総ジョブ数
num_p[1]: 成功したジョブ数

この配列の領域は呼び出し側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 応答情報はジョブ操作APIの応答情報ではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*num_p*が不正(NULL)です。

A.2.3 pjcmd_get_jobresult_info()

```
pjcmd_result_t pjcmd_get_jobresult_info(const PjcmdResp_t *resp_p, pjcmd_jobresult_type_t result, int64_t indx,
 pjcmd_jobresult_info_t type, void *val_p)
```

ジョブ操作(*)の応答情報から、特定のジョブについての結果情報を取得します。
(*)ジョブの投入、削除、固定、固定解除、シグナル送信、パラメーター変更、および終了待ち合わせ("付録B リファレンス:ジョブ操作API"参照)

[引数]

resp_p

応答情報へのポインター

result

結果を取得する対象ジョブの条件

PJCMD_JOBRESULT_ANY

すべてのジョブを対象とします。

PJCMD_JOBRESULT_OK

操作に成功したジョブだけを対象とします。

PJCMD_JOBRESULT_ERR

操作に失敗したジョブだけを対象とします。

indx

結果情報を取得するジョブのインデックス。値は、0から"引数*result*で示された条件のジョブの総数-1"の範囲になければいけません。

type

取得する結果情報の識別子(下記の表を参照)

val_p

val_p*に結果情報を格納します。*type*に応じたサイズの領域を、呼び出す側が用意する必要があります。例えば、val_p*の型がint の場合、int型の領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。

type	**val_p*	**val_p*の型
PJCMD_JOBRESULT_SUBJOBID	ジョブのサブジョブID構造体 得られた値を応答情報の解放後に参照した場合の動作は不定です。	PjcmdSubjobid_t *
PJCMD_JOBRESULT_CODE	ジョブの操作の結果コード 0: 成功 0以外: 失敗	int
PJCMD_JOBRESULT_SUB_CODE	ジョブの操作失敗時の詳細結果コード トラブル時の調査用の情報です。	int
PJCMD_JOBRESULT_DETAIL	ジョブの操作失敗時の詳細情報(文字列) トラブル時の調査用の情報です。	char *
PJCMD_JOBRESULT_PJM_CODE	ジョブ終了コード(PJMコード) ジョブの終了待ち合わせ時だけ意味を持ちます。	int32_t
PJCMD_JOBRESULT_EXIT_CODE	ジョブスクリプトの終了コード ジョブの終了待ち合わせ時だけ意味を持ちます。	int32_t
PJCMD_JOBRESULT_SIGNAL_NUM	ジョブスクリプトがシグナルで終了した場合のシグナル番号 ジョブの終了待ち合わせ時だけ意味を持ちます。	int32_t
PJCMD_JOBRESULT_JOB_STATUS	待ち合わせ時のジョブの状態 0: 終了したステップジョブまたはバルクジョブ 1: ACCEPT、QUEUED、READY、RUNNING-A、RUNNING-P、RUNNING、またはRUNNING-E状態 2: HOLD状態 3: ERROR状態 ジョブの終了待ち合わせ時だけ意味を持ちます。	int32_t
PJCMD_JOBRESULT_ACCEPT_DATE	ジョブの受付時刻 ジョブの終了待ち合わせ時だけ意味を持ちます。サブジョブの場合は、そのサマリジョブの受付時刻を返します。	strcut timespec
PJCMD_JOBRESULT_NODE_MODEL	計算ノードの種類 0: FXサーバ 1: PRIMERGYサーバ	uint32_t

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 応答情報はジョブ操作の応答情報ではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*result*または*type*に不明な値が指定されました。

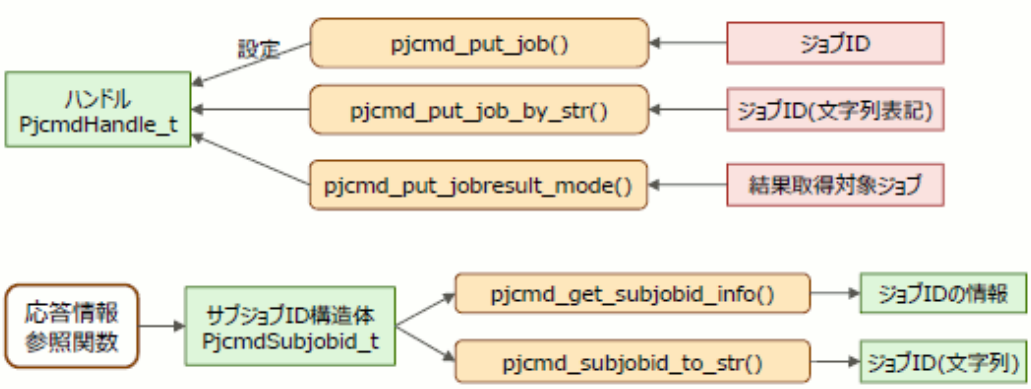
PJCMD_ERROR_NODATA

*indx*が範囲外の値です。

A.3 ジョブIDの設定と取得

ここでは、ジョブIDの操作や参照をするための関数を説明します。

図A.3 ジョブIDの設定と参照



A.3.1 pjcmd_put_job()

```
pjcmd_result_t pjcmd_put_job(PjcmdHandle_t *handle_p, const int64_t *jobid_p, int64_t jobid_s, int64_t jobid_e,
 const int64_t *no_p, int64_t no_s, int64_t no_e, pjcmd_jobmodel_t model)
```

ハンドルに、引数で指定されたジョブを設定します。ハンドルにすでにジョブが設定されている場合は、追加の設定になります。

[引数]

handle_p

ハンドルへのポインター

jobid_p

jobid_p[]は、1つ以上のジョブIDを要素とする配列です。配列の最後の要素は-1でなければいけません。配列が-1で終端しない場合の動作は不定です。*jobid_p*がNULLの場合や配列要素にジョブIDが1つも指定されない場合(*jobid_p*[0]が-1)は、本引数は無視されます。

jobid_s, *jobid_e*

ジョブIDの範囲指定の開始(*jobid_s*)と終了(*jobid_e*)。
jobid_s>*jobid_e*の場合、関数はエラーPJCMD_ERRを返し、pjcmd_errcodeにPJCMD_ERROR_INVALID_PARAMが設定されます。
*jobid_p*でジョブIDが指定された場合、およびジョブIDとして指定できる範囲(0から2147483647)以外の値を指定した場合は、本引数は無視されます。

no_p

ジョブモデルがステップジョブまたはバルクジョブの場合に、ステップ番号またはバルク番号を配列で指定します。配列の最後の要素は-1でなければいけません。配列が-1で終端しない場合の動作は不定です。*no_p*がNULLの場合、*jobid_p*にジョブIDが2つ以上指定された場合、または*model*にPJCMD_JOBMODEL_NORMALが指定された場合は、本引数は無視されます。
配列要素にステップ番号またはバルク番号が1つも指定されない場合(*no_p*[0]が-1)、関数はエラーPJCMD_ERRを返し、pjcmd_errcodeにPJCMD_ERROR_INVALID_PARAMが設定されます。

no_s, *no_e*

ジョブモデルがステップジョブまたはバルクジョブの場合に、ステップ番号またはバルク番号の開始(*no_s*)と終了(*no_e*)を指定します。
no_s>*no_e*の場合、関数はエラーPJCMD_ERRを返し、pjcmd_errcodeにPJCMD_ERROR_INVALID_PARAMが設定されます。
*no_s*と*no_e*に-1を指定した場合、*no_p*でステップ番号またはバルク番号が示された場合、*jobid_p*でジョブIDが2つ以上指定された場合、または*model*にPJCMD_JOBMODEL_NORMALが指定された場合は、本引数は無視されます。

model

引数*no_p*、*no_s*、*no_e*を指定する場合に、それがステップ番号かバルク番号かを区別するために指定します。

PJCMD_JOBMODEL_STEP

引数*no_p*、*no_s*、*no_e*に有効な値が指定された場合は、それらはステップ番号とみなします。

PJCMD_JOBMODEL_BULK

引数*no_p*、*no_s*、*no_e*に有効な値が指定された場合は、それらはバルク番号とみなします。

引数*model*に上記以外のジョブモデル(PJCMD_JOBMODEL_NORMAL、PJCMD_JOBMODEL_MSWK)が指定された場合、または引数*no_p*、*no_s*、*no_e*に無効な値が設定された場合は、サブジョブIDの指定ではないとみなし、ジョブIDを指定する引数*jobid_p*、*jobid_s*、および*jobid_e*だけが有効になります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブIDの指定ができない種類のハンドルです。

PJCMD_ERROR_INVALID_PARAM

引数で指定したパラメーターが不正です。

- *jobid_s*>*jobid_e*
- *no_p*[0]が-1です。

- *no_s>no_e*

PJCMD_ERROR_UNKNOWN_PARAM

引数*model*に不明な値が指定されました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

参考

設定するジョブの種類によって、以下に示す引数を設定してください。以下で示されていない引数は無視されます。

- 1つのジョブID(100)を指定する場合

```
jobid_p[0]=100;
jobid_p[1]=-1;
no_p=NULL;
no_s=-1;
```

- N個のジョブID(100,101,...)を指定する場合

```
jobid_p[0]=100;
jobid_p[1]=101;
...
jobid_p[N-1]=100+N-1;
jobid_p[N]=-1;
```

- ジョブIDの範囲(100-110)を指定する場合

```
jobid_p=NULL;
jobid_s=100;
jobid_e=110;
```

- ステップジョブまたはバルクジョブの1つのサブジョブID(100_1または100[1])を指定する場合

```
jobid_p[0]=100;
jobid_p[1]=-1;
no_p[0]=1;
no_p[1]=-1;
model=PJCMD_JOBMODEL_STEP または PJCMD_JOBMODEL_BULK
```

- ステップジョブまたはバルクジョブのN個のサブジョブID(100_0,100_1,... または100[0],100[1],...)を指定する場合

```
jobid_p[0]=100;
jobid_p[1]=-1;
no_p[0]=0;
no_p[1]=1;
...
no_p[N-1]=N-1;
no_p[N]=-1;
model=PJCMD_JOBMODEL_STEP または PJCMD_JOBMODEL_BULK
```

- ステップジョブまたはバルクジョブのサブジョブIDの範囲(100_0-10または100[0-10])を指定する場合

```
jobid_p[0]=100;
jobid_p[1]=-1;
no_p=NULL;
no_s=0;
```

```
no_e=10;
model=PJCMD_JOBMODEL_STEP または PJCMD_JOBMODEL_BULK
```

A.3.2 pjcmd_put_job_by_str()

```
pjcmd_result_t pjcmd_put_job_by_str(PjcmdHandle_t *handle_p, const char *str_p)
```

ハンドルに、文字列で指定されたジョブIDを設定します。ハンドルにすでにジョブが設定されている場合は、追加されます。

[引数]

handle_p

ハンドルへのポインター

str_p

設定するジョブIDまたはサブジョブID(文字列表記)
ジョブIDは、以下の形式の文字列で指定できます。

- 単一ジョブID (例: "100")
- 単一サブジョブID (例: "200_1"、"300[5]")
- ジョブIDの範囲指定 (例: "10-50")
- サブジョブIDの範囲指定 (例: "200_1-200_5"、"300[5-10]")

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブIDを指定できない種類のハンドルです。

PJCMD_ERROR_INVALID_ARGUMENT

*str_p*がNULLです。

PJCMD_ERROR_INVALID_PARAM

*str_p*で指定されたジョブIDの書式が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

A.3.3 pjcmd_put_jobresult_mode()

```
pjcmd_result_t pjcmd_put_jobresult_mode(PjcmdHandle_t *handle_p, pjcmd_jobresult_mode_t mode)
```

ジョブ運用管理機能から操作結果を取得する際の対象ジョブの範囲をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

mode

ジョブ操作結果取得モード

PJCMD_JOBRESULT_MODE_ALL

ジョブIDを範囲で指定した場合、取得する結果には存在しないジョブも含まれます。

PJCMD_JOBRESULT_MODE_BASIC

ジョブIDを範囲で指定した場合、取得する結果には存在しないジョブ、操作権限がないジョブ、および操作ができない状態のジョブは含まれません。

この関数を呼び出さない場合、ジョブ操作結果取得モードはPJCMD_JOBRESULT_MODE_ALLが適用されます。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ操作のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*mode*が不明な値です。

PJCMD_ERROR_INTERNAL

内部エラー

A.3.4 pjcmd_get_subjobid_info()

```
pjcmd_result_t pjcmd_get_subjobid_info(const PjcmdSubjobid_t *subjobid_p, pjcmd_subjobid_info_t info, void *val_p)
```

サブジョブID構造体のジョブID、ステップ番号、およびバルク番号を取得します。

[引数]

subjobid_p

サブジョブID構造体へのポインター

info

取得する情報を示す識別子(下記の表を参照)

val_p

val_p*に取得した値を格納します。*info*に応じたサイズの領域を呼び出す側が用意する必要があります。例えば、val_p*の型がint64_tの場合、int64_t型の領域を呼び出し側で用意し、それへのポインター(int64_t *)を*val_p*に指定してください。

info	**val_p*	**val_p*の型
PJCMD_SUBJOBID_JOBID	ジョブID	int64_t
PJCMD_SUBJOBID_STEPNO	ステップ番号	int64_t

info	**val_p*	**val_p*の型
	ステップジョブでない場合は-1が格納されます。	
PJCMD_SUBJOBID_BULKNO	バルク番号 バルクジョブでない場合は-1が格納されます。	int64_t

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*subjobid_p*または*val_p*がNULLです。

PJCMD_ERROR_UNKNOWN_PARAM

*info*に不明な値が指定されました。

A.3.5 pjcmd_subjobid_to_str()

```
pjcmd_result_t pjcmd_subjobid_to_str(const PjcmdSubjobid_t *subjobid_p, char *str_p)
```

サブジョブID構造体を表示用の文字列へ変換します。

[引数]

subjobid_p

サブジョブID構造体へのポインター

str_p

*str_p*が示す領域に変換後の文字列が格納されます。領域は呼び出し側で確保する必要があります。格納される文字列の長さは、終端のNULL文字も含めて、最大でPJCMD_MAX_SUBJOBID_STR_LENバイトになります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*subjobid_p*または*str_p*がNULLです。

[例]

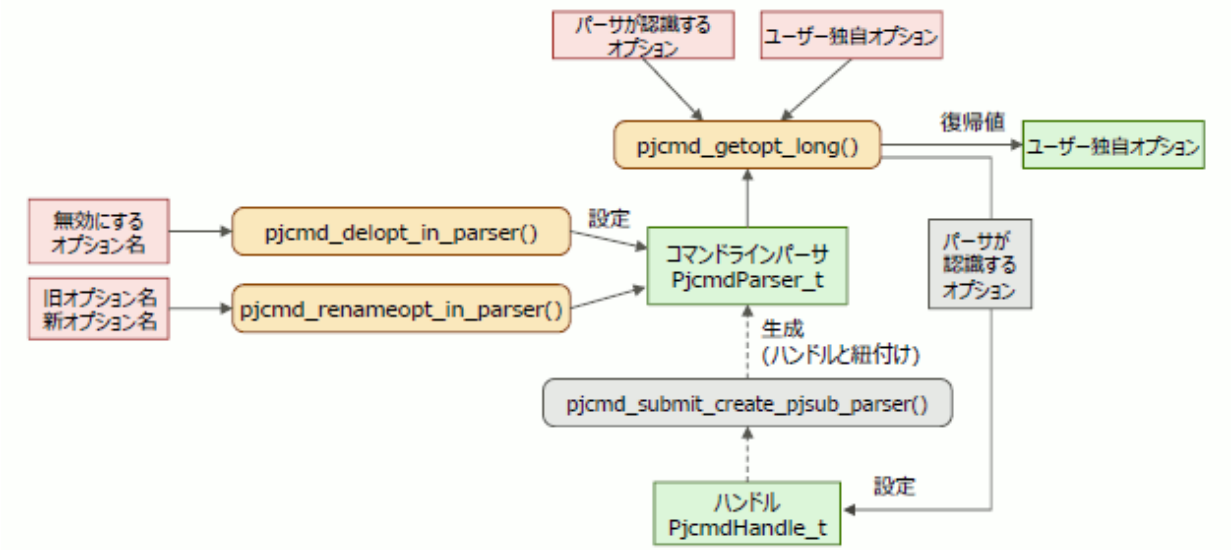
サブジョブID構造体は、そのジョブモデルによって以下のように変換されます。

- 通常ジョブ: "*<ジョブID>*" (例: "123")
- ステップジョブ: "*<ジョブID>*_*<ステップ番号>*" (例: "123_5")
- バルクジョブ: "*<ジョブID>*[*<バルク番号>*]" (例: "123[5]")

A.4 コマンドライン引数の解析

ここでは、コマンドライン引数を解析するための関数を説明します。

図A.4 コマンドライン引数の解析



A.4.1 pjcmd_getopt_long()

```
int pjcmd_getopt_long(const PjcmdParser_t *parser_p, int argc, char *const argv[], const char *optstring_p, const struct option *longopts_p, int *longindex_p)
```

コマンドラインパーサに基づいて、コマンドライン引数を解析します。
この関数の動作は、以下を除いて、関数getopt_long()と同様です。

- 認識するオプションは以下です。
  - 呼び出し側が指定した*optstring_p*(ショートオプション)および*longopts_p*(ロングオプション)
  - コマンドラインパーサ*parser_p*が認識するオプション
- 呼び出し側が指定したオプション(*optstring_p*、*longopt_p*)については、1つ検出するたびに関数は復帰します。コマンドラインパーサ*parser_p*が認識するオプションについては、関数は復帰せず、内部でコマンドラインパーサに対応するハンドル(コマンドラインパーサを生成したときに指定したハンドル)にパラメーターが設定されます。

*parser_p*にNULLが指定された場合、関数getopt_long()と同じ動作になります。

[引数]

parser_p

コマンドラインパーサへのポインター

argc

解析する引数の数

argv

解析する引数の配列

optstring_p

ショートオプション文字列(getopt_long(3)参照)

longopts_p

受け付けるロングオプション(getopt_long(3)参照)

longindex_p

認識したロングオプションへのインデックス(getopt_long(3)参照)

[返り値]

- ショートオプションを検出した場合はその文字を返します。
- ロングオプションを検出した場合は、*longopts_p*内の対応するメンバflagに従った値を返します。
- コマンドラインオプションの解析が終了した場合は、-1を返します。
- 不明なオプションを検出した場合は、'?'を返します。

A.4.2 pjcmd_delopt_in_parser()

```
pjcmd_result_t pjcmd_delopt_in_parser(PjcmdParser_t *parser_p, char opt, const char *longopt_p)
```

コマンドラインパーサが認識するオプションのうち、指定したオプションを無効にします。この関数の1回の呼び出しで、ショートオプションとロングオプションをそれぞれ1つだけ無効にできます。複数のオプションを無効にしたい場合は、この関数を複数回呼び出してください。

[引数]

parser_p

コマンドラインパーサへのポインター

opt

無効にするショートオプション("-"を除いたオプション文字)。NULL文字'¥0'が指定された場合は、無効にするショートオプションは指定されていないとみなします。

longopt_p

無効にするロングオプション名("--"を除いたオプション名)。NULLが指定された場合は、無効にするロングオプションは指定されていないとみなします。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*parser_p*がNULLです。

PJCMD_ERROR_UNKNOWN_OPTION

指定されたオプションはコマンドラインパーサに存在しません。

A.4.3 pjcmd_renameopt_in_parser()

```
pjcmd_result_t pjcmd_renameopt_in_parser(PjcmdParser_t *parser_p, char old_opt, char new_opt, const char *old_longopt_p,
const char *new_longopt_p)
```

コマンドラインパーサが認識するオプションの名前を変更します。

[引数]

parser_p

コマンドラインパーサへのポインター

old_opt

変更前のショートオプション("-"を除いたオプション文字)。ショートオプションを変更しない場合はNULL文字'¥0'を指定してください。

new_opt

変更後のショートオプション("-"を除いたオプション文字)。*old_opt*がNULL文字の場合は無視されます。

old_longopt_p

変更前のロングオプション("--"を除いたオプション名)。ロングオプションを変更しない場合はNULLを指定してください。

new_longopt_p

変更後のロングオプション("--"を除いたオプション名)。*old_longopt_p*がNULLの場合は無視されます。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*parser_p*がNULLです。

PJCMD_ERROR_UNKNOWN_OPTION

指定されたオプションはコマンドラインパーサに存在しません。

PJCMD_ERROR_INVALID_OPTION

変更後のオプションの指定が正しくありません。

- 変更しないほかのオプションと重複しています。
- 変更前のオプション*old_opt*または*old_longopt_p*が指定されているときに、変更後のオプション*new_opt*にNULL文字または*new_longopt_p*にNULLが指定されています。

注意

- この関数はオプションの名前を変更するだけです。オプションに引数が必要か否かの仕様までは変更できません。また、無効にする場合は、関数pjcmd_delopt_in_parser()を使ってください。
- ショートオプションをロングオプションに変更すること、またはその逆はできません。

A.5 usage表示

ここでは、ジョブ運用管理機能が提供するコマンドのusageを表示するための関数を説明します。

図A.5 ジョブ運用管理機能のコマンドのusage表示

A.5.1 pjcmd_print_stdcmd_usage()

```
pjcmd_result_t pjcmd_print_stdcmd_usage(pjcmd_stdcmd_t cmd, const char *cmdname_p)
```

ジョブ運用管理機能が提供するコマンドのusageを標準出力に出力します。

[引数]

cmd

ジョブ運用管理機能が提供するコマンドの識別子

識別子	説明
PJCMD_STDCMD_PJSUB	pjsubコマンド

識別子	説明
PJCMD_STDCMD_PJDEL	pjdelコマンド
PJCMD_STDCMD_PJHOLD	pjholdコマンド
PJCMD_STDCMD_PJRLS	pjrlsコマンド
PJCMD_STDCMD_PJSIG	pjsigコマンド
PJCMD_STDCMD_PJWAIT	pjwaitコマンド
PJCMD_STDCMD_PJALTER	pjalterコマンド
PJCMD_STDCMD_PMALTER	pmalterコマンド
PJCMD_STDCMD_PJSTAT	pjstatコマンド
PJCMD_STDCMD_PJACL	pjaclコマンド
PJCMD_STDCMD_PMPJMOPT	pmpjmoptコマンド
PJCMD_STDCMD_PJSHOWRSC	pjshowrscコマンド

cmdname_p

usage内に表示されるコマンド名。NULLの場合は、ジョブ運用管理機能が提供するコマンド名になります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

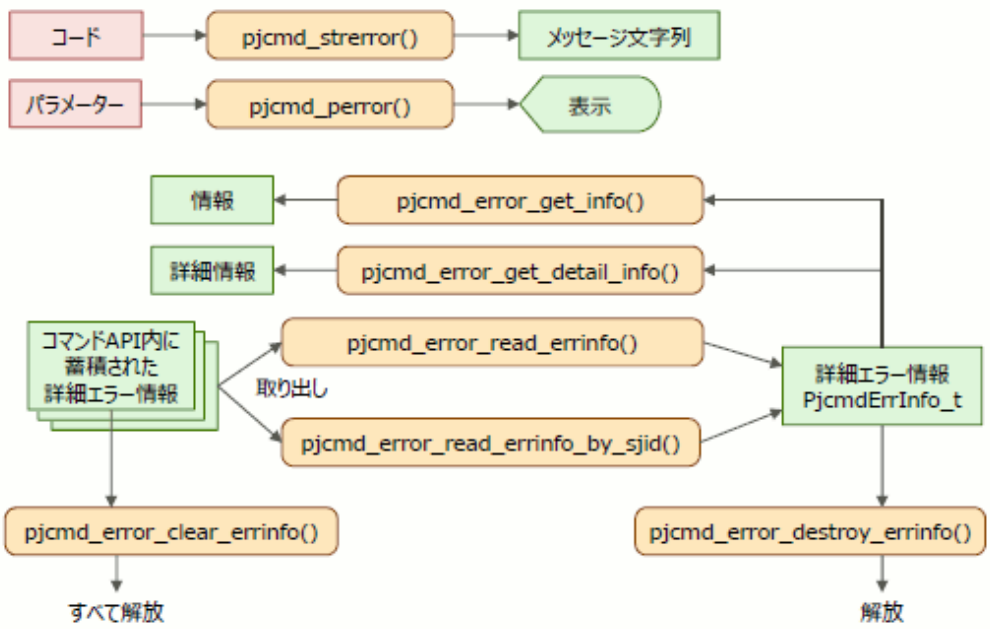
PJCMD_ERROR_UNKNOWN_PARAM

*cmd*に不明な値が指定されました。

A.6 エラー関連

ここでは、コマンドAPIのエラー情報を扱うための関数を説明します。

図A.6 エラー情報の参照



A.6.1 pjcmd_strerror()

```
char *pjcmd_strerror(pjcmd_error_t code)
```

引数で指定されたエラーコードに対応するコマンドAPIの標準エラーメッセージを返します。

[引数]

code

エラーコード。指定できる値は"A.7.3 変数pjcmd_errcode"を参照してください。

[返り値]

*code*に応じて、以下の固定文字列へのポインターを返します。

エラーコード	メッセージ
PJCMD_SUCCESS	Succeeded
PJCMD_ERROR_INVALID_HANDLE	Invalid handle
PJCMD_ERROR_INVALID_RESP	Invalid response data
PJCMD_ERROR_INVALID_ARGUMENT	Invalid argument
PJCMD_ERROR_UNKNOWN_OPTION	Unknown option
PJCMD_ERROR_INVALID_OPTION	Invalid option
PJCMD_ERROR_UNKNOWN_PARAM	Unknown parameter
PJCMD_ERROR_INVALID_PARAM	Invalid parameter
PJCMD_ERROR_NODATA	No data
PJCMD_ERROR_OPEN	Open failed
PJCMD_ERROR_INVALID_NODE	Invalid node
PJCMD_ERROR_NOPERM	No permission
PJCMD_ERROR_CONNECT	Connection failed
PJCMD_ERROR_NOMEM	Not enough memory

エラーコード	メッセージ
PJCMD_ERROR_BUSY	Operation is busy
PJCMD_ERROR_TOO_LONG	Too long
PJCMD_ERROR_NOENT	No such file or directory
PJCMD_ERROR_ACCESS	Permission denied
PJCMD_ERROR_SIGNAL	Interrupted system call
PJCMD_ERROR_INTERNAL	Internal error occurred
そのほかの値*n*	Unknown code *n*

A.6.2 pjcmd_perror()

```
void pjcmd_perror(pjcmd_msg_level_t level, const char *compo_p, int id, const char *cmdname_p, const char *addmsg_p)
```

エラーコードpjcmd_errcodeの値に応じて、以下の書式でメッセージ標準エラー出力に表示します。

[level] compo_p id cmdname_p message addmsg_p

ここで、*message*はエラーコードpjcmd_errcodeに対応した文字列です("A.6.1 pjcmd_strerror()"参照)。

[引数]

level

メッセージのレベル。レベルに対応した以下の文字列がメッセージ内に出力されます。

level	対応する文字列
PJCMD_MSG_INFO	INFO
PJCMD_MSG_NOTICE	NOTE
PJCMD_MSG_WARN	WARN
PJCMD_MSG_ERROR	ERR. (最後にドットが付きます)
PJCMD_MSG_EMERG	EMRG

compo_p

コンポーネント名。終端のNULL文字を除いて5文字以内です。機能名など、任意の文字列が指定できます。

id

メッセージID。4桁の任意の10進数を指定できます。4桁に満たない場合は、上位桁が0で埋められて表示されます。

cmdname_p

メッセージ中に表示されるコマンド名。コマンドAPIを呼び出しているプログラム名など、任意の文字列が指定できます。

addmsg_p

追加メッセージ。NULLを指定した場合、この引数は無視されます。

[例]

```
pjcmd_perror(PJCMD_MSG_ERROR, "MYCMD", 1, "mycommand", "(-x)");
```

pjcmd_errcodeがPJCMD_ERROR_UNKNOWN_OPTIONの場合、出力メッセージは以下になります。

```
[ERR.] MYCMD 0001 mycommand Unknown option (-x)
```

A.6.3 pjcmd_error_read_errinfo()

```
PjcmdErrInfo_t *pjcmd_error_read_errinfo(void)
```

コマンドAPI内に蓄積されている詳細エラー情報のリストから次の詳細エラー情報を取り出します。呼び出すたびに、次の詳細エラー情報が返されます。

[引数]

なし

[返り値]

詳細エラー情報。
取り出す詳細エラー情報が存在しない場合はNULLを返し、pjcmd_errcodeにはエラーコードPJCMD_ERROR_NODATAが設定されます。

[pjcmd_errcode]

PJCMD_ERROR_NODATA

次の詳細エラー情報はありません。

A.6.4 pjcmd_error_read_errinfo_by_sjid()

```
PjcmdErrInfo_t *pjcmd_error_read_errinfo_by_sjid(const PjcmdSubjobid_t *sjid_p)
```

コマンドAPI内に蓄積されている詳細エラー情報のリストから、指定したサブジョブID構造体に対応するジョブの詳細エラー情報を取り出します。呼び出すたびに、該当する次の詳細エラー情報を返します。

[引数]

sjid_p

サブジョブID構造体へのポインター。ジョブの操作がエラーになったときに関数pjcmd_get_jobresult_info()で取得するサブジョブID構造体を指定します。

[返り値]

詳細エラー情報。
失敗した場合はNULLを返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_NODATA

次の詳細エラー情報はありません。または、*sjid_p*に該当する詳細エラー情報はありません。

PJCMD_ERROR_INVALID_ARGUMENT

*sjid_p*が不正(NULL)です。

A.6.5 pjcmd_error_get_info()

```
pjcmd_result_t pjcmd_error_get_info(const PjcmdErrInfo_t *errinfo_p, pjcmd_errinfo_type_t type, void *val_p)
```

詳細エラー情報から、指定した情報項目の値を参照します。

[引数]

errinfo_p

詳細エラー情報へのポインター。関数pjcmd_error_read_errinfo()で取得した情報を指定します。

type

参照したい情報項目の識別子(下記の表を参照)

val_p

val_p*に値を格納します。*type*に応じたサイズの領域を、呼び出す側が用意する必要があります。例えば、val_p*の型がintの場合、int型の領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。

type	**val_p*	**val_p*の型
PJCMD_ERRINFO_SUBERRCODE	詳細エラーコード 詳細エラー情報が示すエラーの種類です。格納される値(PJCMD_SUBERR_*xxxx*)とその意味は"A.7.2 詳細エラーコード"を参照してください。	int
PJCMD_ERRINFO_CODE1 PJCMD_ERRINFO_CODE2 PJCMD_ERRINFO_CODE3	エラーの詳細情報(数値) *type*にPJCMD_ERRINFO_SUBERRCODEを指定して得られた詳細エラーコード(PJCMD_SUBERR_*xxxx*)には、さらに最大で3つの詳細情報(数値)が付属する場合があります("A.7.2 詳細エラーコード"の説明で示されている"[CODE *n*]")。それらを参照する場合に、*type*にPJCMD_ERRINFO_CODE*n*を指定します。 詳細エラーコードが詳細情報(数値)を持たない場合は、得られる値に意味はありません。	int
PJCMD_ERRINFO_SCRIPTNAME	エラーがジョブスクリプト内で発生した場合に、そのジョブスクリプト名 ジョブスクリプトに起因しないエラーの場合、**val_p*はNULLになります。 得られた値を詳細エラー情報の解放後に参照した場合の動作は不定です。	char *
PJCMD_ERRINFO_SCRIPTLINE	ジョブスクリプト内でエラーが発生した行番号 ジョブスクリプトに起因しないエラーの場合、**val_p*は0になります。	int
PJCMD_ERRINFO_SJID	エラーが発生したジョブのサブジョブID構造体 ジョブのジョブID、バルク番号、およびステップ番号を知ることができます。サブジョブIDが関与しないエラーの場合、**val_p*はNULLになります。 得られた値を詳細エラー情報の解放後に参照した場合の動作は不定です。	PjcmdSubjobid_t *
PJCMD_ERRINFO_PLACEID	エラー発生箇所(トラブル時の調査用)	uint64_t

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*errinfo_p*または*val_p*がNULLです。

PJCMD_ERROR_UNKNOWN_PARAM

*type*が不正です。

A.6.6 pjcmd_error_get_detail_info()

```
const char *pjcmd_error_get_detail_info(const PjcmdErrInfo_t *errinfo_p, int indx)
```

1つの詳細エラー情報に含まれる複数の詳細情報(文字列)のうち、指定したインデックスの詳細情報を返します。

[引数]

errinfo_p

詳細エラー情報へのポインター

indx

詳細エラー情報に含まれる詳細情報(文字列)のインデックス。詳細エラーコードによっては最大で5つあり、0から4の値を指定します。
詳細エラーコードに対応した詳細情報については、"A.7.2 詳細エラーコード"を参照してください。

[返り値]

詳細情報(文字列)。
本関数は、*indx*に対応する詳細情報が存在しない場合または失敗した場合にNULLを返します。両者を区別するには、pjcmd_errcode で判定してください。

[pjcmd_errcode]

PJCMD_SUCCESS (かつ、返り値が NULL)

この詳細エラー情報には、*indx*に対応する詳細情報は存在しません。

PJCMD_ERROR_UNKNOWN_PARAM

*indx*が指定できる範囲を超えています。

PJCMD_ERROR_INVALID_ARGUMENT

*errinfo_p*がNULLです。

A.6.7 pjcmd_error_destroy_errinfo()

```
void pjcmd_error_destroy_errinfo(PjcmdErrInfo_t *errinfo_p)
```

指定された詳細エラー情報を解放します。

[引数]

errinfo_p

詳細エラー情報へのポインター

[返り値]

なし

A.6.8 pjcmd_error_clear_errinfo()

```
void pjcmd_error_clear_errinfo(void)
```

コマンドAPI内に蓄積されている詳細エラー情報をすべて解放します。

[引数]

なし

[返り値]

なし

A.7 エラーコード、グローバル変数、および定数

ここでは、コマンドAPIのエラーコード、グローバル変数、および定数(マクロ)を説明します。

A.7.1 結果コード

コマンドAPI関数の呼び出し結果です。

値	説明
PJCMD_OK	成功
PJCMD_ERR	失敗

A.7.2 詳細エラーコード

関数pjcmd_error_read_errinfo()で取得する詳細エラー情報の種類を示す値です。

参考

詳細エラーコードに付随する情報がある場合は、表内に以下の記号で示しています。

[CODE *n*] 関数pjcmd_error_get_info()で詳細コード1～3が取得できます。
[DETAIL *n*] 関数pjcmd_error_get_detail_info()で詳細情報(文字列)0～4が取得できます。
[JOBID] 関数pjcmd_error_get_info()でジョブID(サブジョブID構造体)が取得できます。

詳細エラーコード	説明
PJCMD_SUBERR_UNKNOWN_OPT	不明なオプションを検出しました。 [DETAIL 0] 不明なオプション名
PJCMD_SUBERR_COMBINATION	オプションの組み合わせが正しくありません。 [DETAIL 0] オプション名1 [DETAIL 1] オプション名2
PJCMD_SUBERR_UNKNOWN_OPTARG	オプションの引数が不明な値です。 [DETAIL 0] オプション名 [DETAIL 1] 不明な値の引数
PJCMD_SUBERR_INVALID_OPTARG	オプションの引数に指定できない値が指定されました。 [DETAIL 0] オプション名 [DETAIL 1] 引数または引数=値
PJCMD_SUBERR_NO_PARAM	パラメーターが指定されていません。
PJCMD_SUBERR_ARG_FORMAT_QUOTA	ダブルクォート(")またはシングルクォート(')が対応していません。
PJCMD_SUBERR_INVALID_ARG	引数の指定が間違っています。 [DETAIL 0] 引数
PJCMD_SUBERR_NO_JOBID	ジョブIDが指定されていません。
PJCMD_SUBERR_JOBID_SYNTAX_ERROR	ジョブIDの書式が間違っています。 [DETAIL 0] 間違っているジョブID
PJCMD_SUBERR_JOBID_NOT_EXIST	指定されたジョブIDに該当するジョブは存在しません。 [JOBID]
PJCMD_SUBERR_JOB_STATE_ERROR	指定されたジョブの状態ではコマンドを実行できません。 [JOBID]
PJCMD_SUBERR_JOB_TYPE_ERROR	会話型ジョブに対して、ジョブのパラメーター変更をしようとしました。 [JOBID]
PJCMD_SUBERR_JOB_MODEL_ERROR	ステップジョブに対して、リソースユニット名を変更する際にサブジョブIDを指定しました。
PJCMD_SUBERR_JOBNAME_MISMATCH	ステップジョブのサブジョブをまとめて投入する際に、ジョブ名が一致していません。 [DETAIL 0] ジョブスクリプト名
PJCMD_SUBERR_DUP_REQUEST	すでに操作依頼は受付済みです。 [JOBID]

詳細エラーコード	説明
PJCMD_SUBERR_FILE_OPEN	ファイルのオープンに失敗しました。 [CODE 1] オープンが失敗したときのerrno [DETAIL 0] ファイル名
PJCMD_SUBERR_FILE_NAME_TOO_LONG	ファイル名が長すぎます。 [DETAIL 0] ファイル名
PJCMD_SUBERR_FILE_FORMAT	ファイルの書式が間違っています。 [CODE 1] 行番号 [DETAIL 0] ファイル名
PJCMD_SUBERR_MULTIPLE_SCRIPT	ジョブスクリプトが複数指定されました。
PJCMD_SUBERR_TOO_MANY_ARG_SCRIPT	ジョブスクリプトに記述したオプションが多すぎます。 [DETAIL 0] 解析できなかったオプション名
PJCMD_SUBERR_LINE_LENGTH	ジョブスクリプトの1行に記述できる文字数を超えました。
PJCMD_SUBERR_CURRENT_ACCESS	カレントディレクトリの情報を取得できませんでした。
PJCMD_SUBERR_CURRENT_PATH	カレントディレクトリ名が不正です。または、ディレクトリ名に改行コードが含まれています。
PJCMD_SUBERR_FILE_CREAT_FAIL	指定されたファイルを作成できません。 [CODE 1] 作成が失敗したときのerrno [DETAIL 0] ファイル名
PJCMD_SUBERR_NOT_FOUND_UNAME	指定されたユーザー名は存在しません。 [DETAIL 0] ユーザー名
PJCMD_SUBERR_GID_FAILED	指定されたグループ ID からグループ名を獲得できませんでした。 [DETAIL 0] グループID(文字列)
PJCMD_SUBERR_CANNOT_USED_OPT	使用できないオプションが指定されました。 [DETAIL 0] オプション名
PJCMD_SUBERR_RESOURCE_ERROR	指定した資源は、対象のジョブに対しては正しくありません。 [JOBID] [DETAIL 0] ジョブのパラメーター変更操作で指定した資源情報
PJCMD_SUBERR_OPERATION_BUSY	ほかの要求を処理中のため、要求を実行できません。 [JOBID]
PJCMD_SUBERR_LOWER_LIMIT	指定された資源量が下限値を下回っています。 [JOBID] (情報がない場合もあります) [DETAIL 0] 指定された資源名 [DETAIL 1] 指定された資源量 [DETAIL 2] 指定できる資源量の下限
PJCMD_SUBERR_TIME_ERROR	指定されたジョブの実行開始予定時刻が、現在時刻より過去です。 [DETAIL 0] 時刻 ("*YYYYMMDDhhmm*") *YYYY*:年、*MM*:月、*DD*:日、*hh*:時、*mm*:分
PJCMD_SUBERR_UPPER_LIMIT	指定された資源量が上限値を超えています。 [JOBID] (情報がない場合もあります) [DETAIL 0] 指定された資源名 [DETAIL 1] 指定された資源量 [DETAIL 2] 指定できる資源量の上限
PJCMD_SUBERR_RSC_DOES_NOT_EXIST	指定された資源が存在しません。 [DETAIL 0] 指定された資源名 [DETAIL 1] 指定された資源量
PJCMD_SUBERR_RSC_IS_DISABLED	指定された資源は利用できない状態です。 [DETAIL 0] 指定された資源名 [DETAIL 1] 指定された資源量

詳細エラーコード	説明
PJCMD_SUBERR_STEPNO_OVERFLOWED	指定されたステップ番号が規定された範囲を超えています。
PJCMD_SUBERR_ARCH_ERROR	ジョブの投入されたリソースユニットから、機種が異なるリソースユニットに変更しようとしました。
PJCMD_SUBERR_POLICY_M [PG]	仮想ノード配置ポリシー指定に関して、オプション*opt1*で指定した値*m*は、オプション*opt2*で指定した値*n*の倍数でなければいけません。 [DETAIL 0] オプション名 *opt1* [DETAIL 1] 値*m* [DETAIL 2] オプション名 *opt2* [DETAIL 3] 値*n* pjsubコマンドの-L vnode=*m*相当の指定と-P vn-policy=unpack/abs-unpack=*n*相当の指定をしたときのエラーです。
PJCMD_SUBERR_SYSTEM_CONF_CHANGED	ジョブの受付処理中にシステム構成が変更されました。
PJCMD_SUBERR_RSCNAME_NOT_SPECIFIED	資源名が指定されていません。
PJCMD_SUBERR_NO_EXECUTE_PERMISSION	操作の権限がありません。
PJCMD_SUBERR_NO_JOBID_PERMISSION	指定したジョブに対する操作が許されていません。 [JOBID]
PJCMD_SUBERR_ACCEPT_LIMIT	ジョブの受付数が上限を超えています。 [JOBID] (情報がない場合もあります) [DETAIL 0] 上限を超えた項目名 "ru-accept" : リソースユニットにおけるバッチジョブの同時受付数 "ru-accept-allsubjob" : リソースユニットにおけるバルクジョブおよびステップジョブのサブジョブの同時受付数 "ru-accept-bulksubjob" : リソースユニットにおけるバルクジョブのサブジョブの同時受付数 "ru-accept-stepsubjob" : リソースユニットにおけるステップジョブのサブジョブの同時受付数 "ru-interact-accept" : リソースユニットにおける会話型ジョブの同時受付数 "rg-accept" : リソースグループにおけるバッチジョブの同時受付数 "rg-accept-allsubjob" : リソースグループにおけるバルクジョブおよびステップジョブのサブジョブの同時受付数 "rg-accept-bulksubjob" : リソースグループにおけるバルクジョブのサブジョブの同時受付数 "rg-accept-stepsubjob" : リソースグループにおけるステップジョブのサブジョブの同時受付数 "rg-interact-accept" : リソースグループにおける会話型ジョブの同時受付数
PJCMD_SUBERR_OUT_OF_RANGE	指定された値がジョブACL機能で定義されている範囲を超えています。 [JOBID] (情報がない場合もあります) [DETAIL 0] オプション名
PJCMD_SUBERR_RSCNAME_ILLEGAL	ジョブ投入時に指定可能な資源に関して、ジョブACL機能で定義されたデフォルト値との組み合わせが不正です。 [DETAIL 0] 指定された資源名 [DETAIL 1] 詳細メッセージ
PJCMD_SUBERR_RSC_CANNOT_BE_SPECIFIED	指定されたカスタム資源の値は、指定可能なカスタム資源の種別には含まれていません。 [JOBID] (情報がない場合もあります) [DETAIL 0] 指定されたカスタム資源名 [DETAIL 1] 指定された値 [DETAIL 2] 指定可能な値

詳細エラーコード	説明
PJCMD_SUBERR_TOO_MANY_CUSTOMRSC	指定されたカスタム資源の個数が多すぎます。
PJCMD_SUBERR_NOT_FOUND_CSTMRSC	指定された資源の値に対応するカスタム資源は定義されていません。 [DETAIL 0] 指定された資源名 [DETAIL 1] 指定された値
PJCMD_SUBERR_GATE_CHECK_ERROR	管理者による出口機能の設定でジョブの受付が拒否されました。 [CODE 1] ジョブマネージャー出口で設定されたメッセージID(情報がない場合もあります) [DETAIL 0] ジョブマネージャー出口で設定されたメッセージ(情報がない場合もあります)
PJCMD_SUBERR_JOB_TIMEOUT	会話型ジョブで、ジョブへ割り当てる計算機資源を一定時間内に決定できなかったため、ジョブがキャンセルされました。 [JOBID]
PJCMD_SUBERR_NOT_LOGIN_NODE	会話型ジョブがログインノード以外で実行されました。
PJCMD_SUBERR_JOB_FAIL	会話型ジョブの実行が失敗しました。 [JOBID]
PJCMD_SUBERR_JOB_CANCEL	会話型ジョブの実行がキャンセルされました。 [JOBID]
PJCMD_SUBERR_NOT_SUPPORTED	指定された機能、または機能の組み合わせは、現在、サポートされていません。 [DETAIL 0] 詳細メッセージ
PJCMD_SUBERR_DAEMON_ISNOT_PRESENT	ジョブ運用管理機能が動作していない、またはジョブ運用管理機能と通信ができない状態です。
PJCMD_SUBERR_INTERNAL_ERROR	内部エラー [CODE 1～3] 調査用の情報 [DETAIL 0～4] 調査用の情報
PJCMD_SUBERR_PERMIT	操作する権限がありません。
PJCMD_SUBERR_NODE_ERROR	このノードでは操作できません。
PJCMD_SUBERR_SYSFUNC_STOP	システム管理機能が動作していない、またはシステム管理機能からの情報取得ができない状態です。
PJCMD_SUBERR_NO_CLSTNAME	クラスタ名が指定されていません。
PJCMD_SUBERR_REJECT_OPT	指定したジョブに対する操作が拒否されました。 [JOBID] [DETAIL 0] オプション名 [DETAIL 1] 詳細メッセージ
PJCMD_SUBERR_NOT_EXIST_RSCUNIT	指定されたリソースユニットまたはジョブACL機能で定義されているデフォルトのリソースユニットが存在しません。 [DETAIL 0] リソースユニット名
PJCMD_SUBERR_NOT_EXIST_RSCGRP	指定されたリソースグループまたはジョブACL機能で定義されているデフォルトのリソースグループが存在しません。 [DETAIL 0] リソースグループ名
PJCMD_SUBERR_NO_SIGNO	シグナル送信操作で、シグナルが指定されていません。
PJCMD_SUBERR_NOT_FOUND_GROUP	ジョブACL情報取得操作で、指定したグループは存在しません。 [DETAIL 0] グループ名
PJCMD_SUBERR_NO_USER_PERMISSION	ジョブACL情報取得APIで、指定したユーザーまたはグループに対する表示が許可されていません。 [DETAIL 0] ユーザー名 [DETAIL 1] グループ名

詳細エラーコード	説明
PJCMD_SUBERR_GROUP_NOT_AUTHORIZED	ジョブ投入APIで、このグループの権限では、ジョブ投入は許可されていません。 [DETAIL 0] グループ名
PJCMD_SUBERR_NO_EXEC_PERM_OPT	オプションの指定が許可されていません。 [DETAIL 0] オプション名 これは、示されるオプション名に相当する操作をしたときのエラーです。
PJCMD_SUBERR_NO_EXEC_PERM_JIDOPT	ジョブに対する操作またはオプションの指定が許可されていません。 [JOBID] [DETAIL 0] オプション名 これは、示されるオプション名に相当する操作をしたときのエラーです。
PJCMD_SUBERR_NO_EXEC_PERM_JID	ジョブに対する操作が許可されていません。 [JOBID]
PJCMD_SUBERR_DAEMON_INTERNAL	ジョブ運用管理機能のデーモンの内部エラー [CODE 1] 調査用の情報1 [CODE 2] 調査用の情報2
PJCMD_SUBERR_DAEMON_INTERNAL_RETVAL	ジョブ運用管理機能のデーモンの内部エラー [CODE 1] 調査用の情報
PJCMD_SUBERR_SIGNAL	シグナルを受信しました。
PJCMD_SUBERR_CMD_UNAVAILABLE	要求された操作はできません。
PJCMD_SUBERR_NODATA	該当する情報はありません。 [DETAIL 0] 指定された情報の項目などを示す文字列
PJCMD_SUBERR_DUP_OPT	同じオプションが複数回指定されています。 [DETAIL 0] オプション名
PJCMD_SUBERR_FEW_OPTION	同時に指定すべきオプションが設定されていません。 [DETAIL 0] オプション名1 [DETAIL 1] オプション名2 これは、示される2つのオプション名に相当する操作をしていない場合のエラーです。
PJCMD_SUBERR_CONFLICT	同時に指定できないオプションが指定されました。
PJCMD_SUBERR_INVAL_RUNITNAME	指定したリソースユニット名が不正です。 [DETAIL 0] リソースユニット名
PJCMD_SUBERR_INVAL_ID	指定したノードグループID、ブートグループID、またはノードIDが不正です。 [DETAIL 0] 指定したID
PJCMD_SUBERR_INVAL_RANGE	パラメーターの範囲指定が不正です。 [DETAIL 0] 指定したパラメーターの範囲
PJCMD_SUBERR_INVAL_RGRPNAME	指定したリソースグループ名が不正です。 [DETAIL 0] リソースグループ名
PJCMD_SUBERR_SAME_ID	指定したノードグループID、ブートグループID、またはノードIDに同じものがあります。 [DETAIL 0] 指定したID
PJCMD_SUBERR_SAME_RUNAME	指定したリソースユニット名に同じものがあります。 [DETAIL 0] リソースユニット名
PJCMD_SUBERR_SAME_RGNAME	指定したリソースグループ名に同じものがあります。 [DETAIL 0] リソースグループ名

詳細エラーコード	説明
PJCMD_SUBERR_NOT_FOUND_ID	指定したノードグループID、ブートグループID、またはノードIDに存在しないものがあります。 [DETAIL 0] 指定したID
PJCMD_SUBERR_NOT_FOUND_RUNAME	指定したリソースユニットが存在しません。 [DETAIL 0] リソースユニット名
PJCMD_SUBERR_PERM	指定したノードの状態を取得する権限がありません。 [DETAIL 0] ノードID
PJCMD_SUBERR_PERM_RUNIT	指定したリソースユニットに対する情報取得権限がありません。 [DETAIL 0] リソースユニット名
PJCMD_SUBERR_NOINFO_NODE	指定したノードの情報がありません。 [DETAIL 0] ノードID
PJCMD_SUBERR_NOINFO_RUNIT	指定したリソースユニットの情報がありません。 [DETAIL 0] リソースユニット名
PJCMD_SUBERR_NOINFO_RGRP	指定したリソースグループの情報がありません。 [DETAIL 0] リソースグループ名
PJCMD_SUBERR_CONNECT	計算クラスタ管理ノードとの通信が失敗しました。 [DETAIL 0] クラスタ名
PJCMD_SUBERR_STANDBYNODE	待機系ノードではこの要求はできません。
PJCMD_SUBERR_WARN_CONNECT	特定の計算クラスタ管理ノードとの通信が失敗しました。 ただし、ほかのクラスタ管理ノードとの通信は成功しています。 [DETAIL 0] クラスタ名
PJCMD_SUBERR_MEMORY	メモリの獲得が失敗しました。 [CODE 1] 失敗時のerrno [CODE 2] 獲得サイズ
PJCMD_SUBERR_PSM_ERROR	内部エラー [CODE 1] 調査用の情報 [DETAIL 0] 調査用の情報
PJCMD_SUBERR_TRN_ERROR	内部エラー [CODE 1] 調査用の情報 [DETAIL 0] 調査用の情報
PJCMD_SUBERR_API_ERROR	内部エラー
PJCMD_SUBERR_SYSCALL_ERROR	内部エラー [CODE 1] 調査用の情報 [DETAIL 0] 調査用の情報
PJCMD_SUBERR_JACCTDB_ERROR	内部エラー [CODE 1] 調査用の情報 [CODE 2] 調査用の情報 [DETAIL 0] 調査用の情報

A.7.3 変数pjcmd_errcode

```
extern __thread pjcmd_error_t pjcmd_errcode
```

コマンドAPIのエラーコードが格納されるグローバル変数です。最後に呼び出したコマンドAPI関数のエラーが設定されます。エラーコードの意味は以下のとおりです。

エラーコード	主な意味
PJCMD_SUCCESS	成功

エラーコード	主な意味
PJCMD_ERROR_INVALID_HANDLE	指定されたハンドルが不正です。 ・ ハンドルへのポインターがNULLです。 ・ 操作の種類が違うハンドルです。
PJCMD_ERROR_INVALID_RESP	指定された応答情報が不正です。 ・ 応答情報へのポインターがNULLです。 ・ 操作の種類が違う応答情報です。
PJCMD_ERROR_INVALID_ARGUMENT	関数の引数が不正です。
PJCMD_ERROR_UNKNOWN_OPTION	不明なオプションが指定されました。主に、引数解析関数でのエラーです。
PJCMD_ERROR_INVALID_OPTION	オプションの指定方法が不正です。主に、引数解析関数でのエラーです。
PJCMD_ERROR_UNKNOWN_PARAM	不明なパラメーターが指定されました。
PJCMD_ERROR_INVALID_PARAM	パラメーターの値が不正です。 ・ 指定方法が間違っています。 ・ 必須のパラメーターが設定されていません。
PJCMD_ERROR_NODATA	該当するデータはありません。 指定したインデックスが範囲外である場合や、それ以上取得するデータがない場合のエラーです。
PJCMD_ERROR_OPEN	ファイルのオープンに失敗しました。
PJCMD_ERROR_INVALID_NODE	関数を呼び出したノードが不適切です。 操作依頼関数pjcmd_*operation*_execute()を呼び出したノードが不適切な場合のエラーです。
PJCMD_ERROR_NOPERM	関数の呼び出し権限がありません。 ・ 管理者ではありません。 ・ ジョブACL機能で呼び出しが制限されています。
PJCMD_ERROR_CONNECT	ジョブ運用管理機能との通信に失敗しました。
PJCMD_ERROR_NOMEM	十分な空きメモリがありません。
PJCMD_ERROR_BUSY	ビジー状態です。 例えば、ある操作依頼関数pjcmd_*operation*_execute()の処理中にほかの操作依頼関数を呼び出した場合です。
PJCMD_ERROR_TOO_LONG	データのサイズが長すぎます。
PJCMD_ERROR_NOENT	ファイルまたはディレクトリが存在しません。
PJCMD_ERROR_ACCESS	ファイルまたはディレクトリに対するアクセス権がありません。
PJCMD_ERROR_SIGNAL	シグナルなどの割り込みにより処理が中断されました。
PJCMD_ERROR_INTERNAL	内部エラーが発生しました。

A.7.4 変数pjcmd_optarg

```
extern __thread char *pjcmd_optarg
```

関数pjcmd_getopt_long()で解析中の引数(*argv*の要素)へのポインター
※関数getopt_long(3)における、optargに相当します。

A.7.5 変数pjcmd_optind

```
extern __thread int pjcmd_optind
```

関数pjcmd_getopt()が次に処理する引数のインデックス
※関数getopt_long(3)における、optindに相当します。

A.7.6 変数pjcmd_optopt

```
extern __thread int pjcmd_optopt
```

関数pjcmd_getopt_long()が認識できないオプションを見つけた場合に、そのオプションが格納されます。
※関数getopt_long(3)におけるoptoptに相当します。

A.7.7 定数PJCMD_UNLIMITED

```
#define PJCMD_UNLIMITED (~0UL)
```

制限値を指定する関数で、無制限(制限をしない)であることを意味する値。

A.7.8 定数PJCMD_UNDEFINED

```
#define PJCMD_UNDEFINED (~1UL)
```

制限値を指定する関数またはジョブACL機能の設定情報を取得する関数で、無効値(指定なし)であることを意味する値。

A.7.9 定数PJCMD_MAX_SUBJOBID_STR_LEN

```
#define PJCMD_MAX_SUBJOBID_STR_LEN 32
```

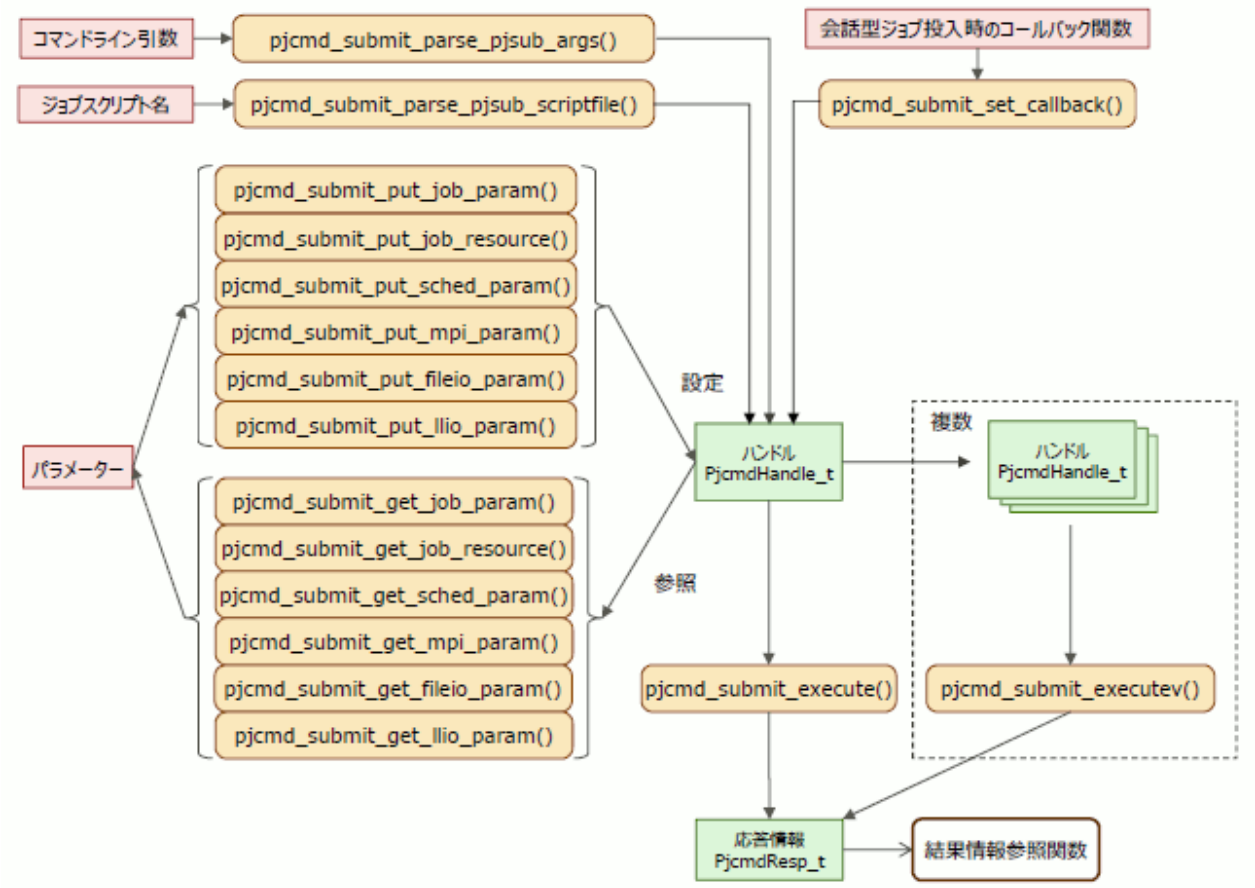
サブジョブID構造体を関数pjcmd_subjobid_to_str()で文字列に変換して格納する際に必要な領域サイズ。終端のNULL文字を含みます。

付録B　リファレンス：ジョブ操作API

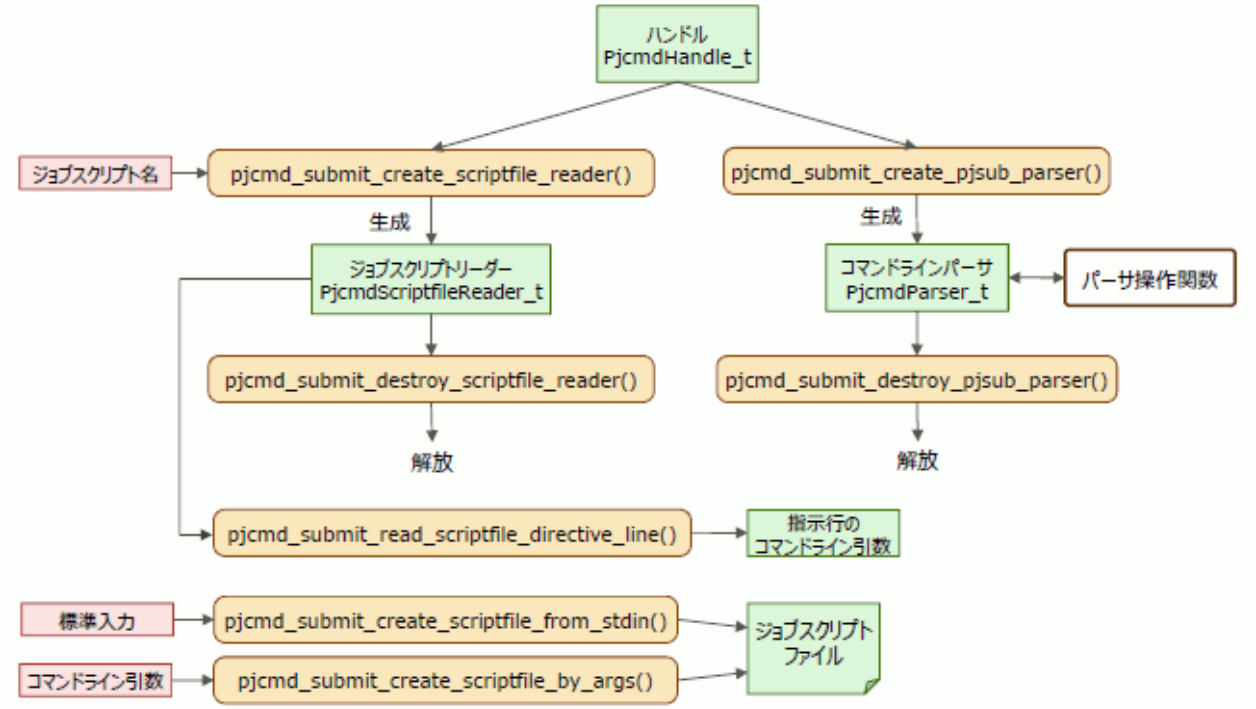
B.1　ジョブの投入

ここでは、ジョブの投入をするための関数を説明します。

図B.1 ジョブの投入依頼



図B.2 ジョブの投入に関するパーサやリーダの操作



B.1.1 pjcmd_submit_parse_pjsub_args()

```
pjcmd_result_t pjcmd_submit_parse_pjsub_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjsubコマンドのオプション仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が不正です。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。
- 排他のオプションが指定されています。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションとそのパラメーターでない引数、すなわちジョブスクリプト名は配列*argv_pp*[]の後ろに移動します。
正常に終了した場合は、変数pjcmd_optindがジョブスクリプト(オプション以外の最初の引数)を指します。呼び出し側は、ジョブスクリプトを別途ハンドルに設定する必要があります。ステップジョブの場合で、2つ以上のジョブスクリプトが指定されている場合は、ハンドルを複製、または新規生成し、ジョブスクリプトごとにハンドルを用意する必要があります。
オプションやそのパラメーターに、不明なものや指定方法が間違っているものを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1]がそのオプションを指します。

B.1.2 pjcmd_submit_parse_pjsub_scriptfile()

```
pjcmd_result_t pjcmd_submit_parse_pjsub_scriptfile(PjcmdHandle_t *handle_p, const char *filename_p, const char
*directive_prefix_p, int32_t *lineno_p, char **detail_pp)
```

ジョブスクリプトファイルを読みながら、その中の指示行をpjsubコマンドのオプション仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

filename_p

ジョブスクリプトのパス

directive_prefix_p

指示行として認識する文字列

lineno_p

**lineno_p*に、エラーを検出した行番号が格納されます。

detail_pp

detail_pp*が、エラーを検出したオプションまたはオプションの引数を示します。detail_pp*が示す領域は、ハンドル内に確保された領域であり、呼び出し側で直接解放しないでください。この領域は、ハンドルの解放または本関数を再度呼び出すまでは保持されます。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。
また、*lineno_p*にエラーを検出した行番号が格納され、**detail_pp* はエラーを検出したオプションまたはその引数を指します。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULL
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

ハンドル以外の引数に不正なもの(NULL)があります。

PJCMD_ERROR_UNKNOWN_OPTION

指示行に不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

指示行に登場するオプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。

B.1.3 pjcmd_submit_create_pjsub_parser()

```
PjcmdParser_t *pjcmd_submit_create_pjsub_parser(PjcmdHandle_t *handle_p)
```

ジョブ投入操作用のコマンドラインパーサを生成します。
このコマンドラインパーサはpjsubコマンドのオプションと同じオプションの仕様情報を保持し、関数pjcmd_getopt_long()で独自のオプション解析をするときに使用します。コマンドラインパーサの情報を変更することで、オリジナルのオプション仕様をカスタマイズできます。

[引数]

handle_p

ハンドルへのポインター

[返り値]

コマンドラインパーサを返します。返されたコマンドラインパーサは、呼び出し側が関数pjcmd_submit_destroy_pjsub_parser()で解放する必要があります。
エラー時はNULLを返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.1.4 pjcmd_submit_destroy_pjsub_parser()

```
pjcmd_result_t pjcmd_submit_destroy_pjsub_parser(PjcmdParser_t *parser_p)
```

ジョブ投入操作用のコマンドラインパーサを解放します。

[引数]

parser_p

解放するコマンドラインパーサへのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*parser_p*が不正(NULL)です。

B.1.5 pjcmd_submit_create_scriptfile_reader()

```
PjcmdScriptfileReader_t *pjcmd_submit_create_scriptfile_reader(const PjcmdHandle_t *handle_p, const char *filename_p,
const char *directive_prefix_p)
```

ジョブスクリプトファイルを読むためのデータ(リーダ)を生成します。

[引数]

handle_p

ハンドルへのポインター

filename_p

ジョブスクリプトのパス

directive_prefix_p

指示行として認識する文字列

[返り値]

ジョブスクリプトファイルのリーダを返します。生成したリーダは、呼び出し側が関数pjcmd_submit_destroy_scriptfile_reader()を利用して解放する必要があります。
エラー時はNULLを返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULL
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

ハンドル以外の引数に不正(NULL)なものがあります。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.1.6 pjcmd_submit_destroy_scriptfile_reader()

```
pjcmd_result_t pjcmd_submit_destroy_scriptfile_reader(PjcmdScriptfileReader_t *reader_p)
```

ジョブスクリプトファイルのリーダを解放します。

[引数]

reader_p

ジョブスクリプトファイルのリーダへのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*reader_p*が不正(NULL)です。

B.1.7 pjcmd_submit_read_scriptfile_directive_line()

```
pjcmd_result_t pjcmd_submit_read_scriptfile_directive_line(PjcmdScriptfileReader_t *reader_p, int *argc_p, char ***argv_ppp)
```

ジョブスクリプトファイルのリーダを利用し、ジョブスクリプトファイルの指示行を1行読み込み、得られるコマンドライン引数を返します。この関数では、リーダに対応するハンドルに設定されている変数(関数pjcmd_submit_put_param()で指定されたもの)の展開や、ダブルクォートなどの特殊文字の解析が行われます。読み込んだ行がコメント行以外("#"で始まらない行)の場合、以降の指示行はコメント行とみなします。改行文字を含む1行の長さが4096文字を超える場合、または指示行に記述されたコマンドライン引数が64個を超える場合はエラーとなります。
本関数は、ジョブスクリプトファイルの指示行を読むだけで、指示行の内容をハンドルに設定することはしません。呼び出し側で引数を独自に解析し、自身でハンドルに値を設定したい場合に利用します。

[引数]

reader_p

ジョブスクリプトファイルのリーダへのポインター

argc_p

**argc_p*に読み込んだ引数の数が格納されます。

argv_ppp

読み込んだ引数が配列(**argv_ppp*)[]として格納されます。
配列の最初の要素*(**argv_ppp*)[0]は、指示行を示す文字列(例:"#PJM")を示します。**argv_ppp*が指す配列の内容は、次の指示行を読むと不定となります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

reader_p、*argc_p*、または*argv_ppp*が不正(NULL)です。

PJCMD_ERROR_TOO_LONG

読み込んだ行の長さが4096文字を超えています。または、1行内の引数が64個を超えています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_NODATA

読み込む指示行がありません。またはすべて読み込みました。

PJCMD_ERROR_INVALID_PARAM

スクリプトファイルの記述が不正です。

PJCMD_ERROR_OPEN

スクリプトファイルの読み込みに失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

B.1.8 pjcmd_submit_put_param()

```
pjcmd_result_t pjcmd_submit_put_param(PjcmdHandle_t *handle_p, pjcmd_submit_param_t param, const void *val_p)
```

ジョブ投入操作のためのパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がchar *型の場合、char *型の値を格納した領域を呼び出し側で用意し、それへのポインター(char **)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_SUBMIT_SCRIPTFILE	ジョブスクリプトのパス(1ファイル) 未設定時は標準入力からジョブの内容を読み込みます。	char *
PJCMD_SUBMIT_JOBNAME	ジョブ名(pjsub -N相当) 未設定時はジョブスクリプト名がジョブ名になります。ジョブスクリプトが標準入力の場合、ジョブ名は "STDIN" になります。	char *
PJCMD_SUBMIT_COMMENT	コメント文字列(pjsub --comment相当)	char *
PJCMD_SUBMIT_JOBMODEL	ジョブモデル PJCMD_JOBMODEL_NORMAL: 通常ジョブ(デフォルト) PJCMD_JOBMODEL_BULK: バルクジョブ PJCMD_JOBMODEL_STEP: ステップジョブ PJCMD_JOBMODEL_MSWK: マスタ・ワーカ型ジョブ	int
PJCMD_SUBMIT_JOBTYPE	ジョブタイプ PJCMD_JOBTYPE_BATCH: バッチジョブ(デフォルト) PJCMD_JOBTYPE_INTERACTIVE: 会話型ジョブ	int
PJCMD_SUBMIT_GNAME	ジョブ実行時のグループ名(pjsub --gname相当) パラメーターPJCMD_SUBMIT_GNAMEとPJCMD_SUBMIT_GIDは最後に指定されたほうが	char *

param	**val_p*	**val_p*の型
	有効になります。 未設定時はカレントグループ名が適用されます。	
PJCMD_SUBMIT_GID	ジョブ実行時のグループID (pjsub --gid相当) パラメーターPJCMD_SUBMIT_GNAMEと PJCMD_SUBMIT_GIDは最後に指定されたほうが有効になります。 未設定時はカレントグループIDが適用されます。	gid_t
PJCMD_SUBMIT_MAILOPT	ジョブのステータスなどの情報について、メール通知が行われる契機と通知内容(pjsub -m相当) b: ジョブ実効開始時(begin) e: ジョブ終了時(end) r: ジョブ再実行時(restart) s: ジョブ統計情報を通知します(ノードごとの情報なし) S: ジョブ統計情報を通知します(ノードごとの情報あり) 未設定時は、メール通知は行われません。	char *
PJCMD_SUBMIT_MAILLIST	ジョブのメール通知の送信先メールアドレス(pjsub --mail-list相当) コンマ(,)で区切って複数指定できます。 未設定時はジョブを投入したユーザーにメールが送信されます。	char *
PJCMD_SUBMIT_WAITMODE	ジョブ投入時の待ち合わせモード(pjsub -w相当) ・ PJCMD_SUBMIT_WAITMODE_WAIT ジョブの投入が完了するまで待ち合わせます(デフォルト)。 (pjsub -wオプションの指定なしに相当) ・ PJCMD_SUBMIT_WAITMODE_JOBCHK ジョブの受付およびチェックが完了するまで待ち合わせます。ただし、ジョブの投入完了までは待ち合わせません。(pjsub -w jobchk相当) ・ PJCMD_SUBMIT_WAITMODE_NOWAIT ジョブの受付が完了するまで待ち合わせます。ジョブのチェックおよび投入は待ち合わせません。(pjsub -w nowait相当)	int
PJCMD_SUBMIT_ENV_INHERIT	全環境変数を計算ノードへ転送するか否かの指定(pjsub -X相当) 0: 転送しない(デフォルト)。 1: 転送する。	int
PJCMD_SUBMIT_BULK_STARTNO	バルクジョブのバルク開始番号(pjsub --bulk --sparam *m*-*n*相当) バルクジョブを投入する場合は必須の設定です。指定できる値の範囲は、0～999999です。	uint32_t
PJCMD_SUBMIT_BULK_ENDNO	バルクジョブのバルク終了番号(pjsub --bulk --sparam *m*-*n*相当) バルクジョブを投入する場合は必須の設定です。指定できる値の範囲は、0～999999です。	uint32_t

param	**val_p*	**val_p*の型
PJCMD_SUBMIT_STEP_DEPEND	ステップジョブ依存関係式(pjsub --step --sparam sd=*form*相当) 書式はpjsubコマンドと同じです(pjsub(1)を参照)。	char *
PJCMD_SUBMIT_STEP_NO	ステップジョブのステップ番号(pjsub --step --sparam sn=*n*相当) 指定できる値の範囲は、0～65534です。	uint16_t
PJCMD_SUBMIT_STEP_JOBNAME	既存ステップジョブのジョブ名(pjsub --step --sparam jnam=*name*相当) パラメーターPJCMD_SUBMIT_STEP_JIDの設定とは排他です。先にPJCMD_SUBMIT_STEP_JIDが設定されている場合はエラーになります。	char *
PJCMD_SUBMIT_STEP_JID	既存ステップジョブのジョブID(pjsub --step --sparam jid=*jobid*相当) パラメーターPJCMD_SUBMIT_STEP_JOBNAMEの設定とは排他です。先にPJCMD_SUBMIT_STEP_JOBNAMEが設定されている場合はエラーになります。 指定できる値の範囲は、0～2147483647です。	uint32_t
PJCMD_SUBMIT_INTERACT_WAITTIME	会話型ジョブの資源割り当て待ち時間(秒) (pjsub --interact --sparam wait-time=*time*相当) 0から36000 までの値、および PJCMD_UNLIMITEDが指定できます。未設定時は0が適用されます。 バッチジョブに対して設定した場合は、関数pjcmd_submit_execute()がエラーになります。	uint64_t
PJCMD_SUBMIT_FSNAME	ファイルシステム名などの任意の文字列(pjsub --fs相当) 設定しない場合、環境変数PJM_FSNAMEが設定されていれば、その値が使われます。	char *
PJCMD_SUBMIT_APPNAME	アプリケーション名などの任意の文字列(pjsub --appname相当) 設定しない場合、環境変数PJM_APPNAMEが設定されていれば、その値が使われます。	char *
PJCMD_SUBMIT_ENV	ジョブ実行時に設定する環境変数(pjsub -x相当) 各要素は"*変数名=値*"の文字列とし、最後の要素はNULLでなければいけません。	char **
PJCMD_SUBMIT_VAR	スクリプトファイルの指示行解析時に利用される変数(pjsub --vset相当) 各要素は"*変数名=値*"の文字列とし、最後の要素はNULLでなければいけません。	char **
PJCMD_SUBMIT_PREFIX	ジョブスクリプト内でディレクティブプレフィックス(pjsub -C相当) 指示行を示す"#PJM"のような文字列です。設定しない場合は、"#PJM"が適用されます。	char *
PJCMD_SUBMIT_SCRIPT_DELIMITER	ステップジョブの複数のジョブスクリプトを区切る文字(pjsub --script-delimiter相当)。	char *

param	**val_p*	**val_p*の型
	未設定時は、区切り文字にはコンマ(,)が使用されます。このパラメーターは、ジョブの投入操作には影響しません。	
PJCMD_SUBMIT_VERBOSE	pjsubコマンドの--verboseオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、ジョブの投入操作には影響しません。	int
PJCMD_SUBMIT_Z	pjsubコマンドの-zオプションの指定に相当。 指定する値は、pjsubコマンドの以下に相当します。 "jid": -z jid オプションの指定。 それ以外の文字列: -z オプションだけの指定。 このパラメーターは、ジョブの投入操作には影響しません。	char *
PJCMD_SUBMIT_NET_ROUTE [FX]	FXサーバでTofuインターコネクトのリンクダウンが発生したときにジョブの実行を継続するかどうかの指定(pjsub --net-routeオプション相当)。 ・ PJCMD_SUBMIT_NET_ROUTE_DYNAMIC Tofuインターコネクトの通信経路を変更します(pjsub --net-route dynamic相当)。ジョブの実行は継続します。 ・ PJCMD_SUBMIT_NET_ROUTE_STATIC Tofuインターコネクトの通信経路は変更しません(pjsub --net-route static相当)。ジョブは異常終了します。 未設定時は、ジョブACL機能の設定に従います。	int
PJCMD_SUBMIT_HELP	pjsubコマンドの--helpオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、ジョブの投入操作には影響しません。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.1.9 pjcmd_submit_get_param()

```
pjcmd_result_t pjcmd_submit_get_param(const PjcmdHandle_t *handle_p, pjcmd_submit_param_t param, void *val_p)
```

ジョブ投入用のハンドルに設定されているパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_submit_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.1.10 pjcmd_submit_put_job_resource()

```
pjcmd_result_t pjcmd_submit_put_job_resource(PjcmdHandle_t *handle_p, const char *rscname_p, const void *val_p)
```

ジョブに割り当てる資源をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

rscname_p

設定する資源名(下記の表を参照)

val_p

設定する資源の量が格納された領域へのポインター。例えば、資源が"node"の場合、ノード形状を表す文字列*X*x*Y*x*Z*"を格納した領域を指すポインター(char *)shape_pを用意し、それへのポインター(char **)&shape_pを*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

rscname_p	**val_p*	**val_p*の型
"node"	ノード数またはノード形状 (pjsub -L node相当) ・ 1次元形状 "*M*[:no-io-exclusive|:io-exclusive][:torus|:mesh|:noncont]" ・ 2次元形状 "*X*x*Y*[:no-io-exclusive|:io-exclusive][:torus|:mesh|:noncont]" ・ 3次元形状 "*X*x*Y*x*Z*[:no-io-exclusive|:io-exclusive][:strict|:strict-io] [:torus|:mesh|:noncont]" ※書式は、pjsub コマンドと同様です。	char *
"vnode"	仮想ノード数 (pjsub -L vnode相当) 指定できる値の範囲は、1～2147483647です。	uint64_t
"node-mem"	node指定時に、ノード当たりのメモリ量上限 (pjsub -L node-mem相当) 指定できる値の範囲は、1048576(1Mi)～2251799812636672(2147483647Mi)Byteです。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	size_t
"elapse"	経過時間制限値 (pjsub -L elapse=*limit*相当) 指定できる値の範囲は、1～2147483647秒です。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	time_t
"adaptive-elapse"	ジョブの経過時間の最小値と最大値 (pjsub -L elapse=*min*-*max*相当) time_t 型の2要素を持つ配列*elapse*[]で指定します。 (time_t)*elapse*[0] : 経過時間の最小値 (time_t)*elapse*[1] : 経過時間の最大値 指定できる値の範囲は、1～2147483647秒です。ただし、*elapse*[0]は*elapse*[1]より小さくなければいけません。 ジョブの経過時間が*elapse*[0]を超えた後の実行可能時間を制限しない場合(pjsub -L elapse=*min*-unlimited に相当)は、*elapse*[1]にPJCMD_UNLIMITEDを指定してください。*elaspe*[1]にPJCMD_UNDEFINEDを指定した場合(pjsub -L elapse=*min*- に相当)は、ジョブACL機能で設定されている最大値が適用されます。 なお、*elapse*[0]にPJCMD_UNLIMITEDまたはPJCMD_UNDEFINEDを指定した場合は、エラーになります。 資源名"elapse"と"adaptive-elapse"は最後に指定されたものが有効になります。	time_t *
"vnode-core"	"vnode"指定時の仮想ノード当たりのCPUコア数(pjsub -L vnode-core相当) 指定できる値の範囲は、1～2147483647です。	unit64_t
"core-mem"	"vnode"指定時のCPUコア当たりの使用メモリ量上限(pjsub -L core-mem相当) 指定できる値の範囲は、1048576 (1Mi)～2251799812636672 (2147483647Mi) Byteです。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	size_t
"vnode-mem"	"vnode"指定時に、1つの仮想ノード当たりの使用メモリ量上限(pjsub -L vnode-mem相当)	size_t

rscname_p	**val_p*	**val_p*の型
	指定できる値の範囲は、1048576 (1Mi)～2251799812636672 (2147483647Mi) Byteです。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	
"rscunit"	ジョブを投入するリソースユニット名(pjsub -L rscunit、-L ru相当)	char *
"rscgrp"	ジョブを投入するリソースグループ名(pjsub -L rscgrp、-L rg相当)	char *
"proc-core"	プロセス単位のコアファイルサイズ制限(pjsub -L proc-core相当) 指定できる値の範囲は、0～2147483647バイト(2GiB-1)です。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	size_t
"proc-cpu"	プロセス単位のCPU時間制限(pjsub -L proc-cpu相当) 指定できる値の範囲は、1～2147483647です。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	uint64_t
"proc-crproc"	プロセス単位のプロセス生成数制限(pjsub -L proc-crproc相当) 指定できる値の範囲は、0～2147483647です。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	uint64_t
"proc-data"	プロセス単位のデータセグメントサイズ制限(pjsub -L proc-data相当) 指定できる値の範囲は、0～2251799812636672 (2147483647Mi) Byteです。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	size_t
"proc-lockm"	プロセス単位のロックメモリサイズ制限(pjsub -L proc-lockm相当) 指定できる値の範囲は、0～2251799812636672 (2147483647Mi) Byteです。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	size_t
"proc-msgq"	プロセス単位のメッセージキューサイズ制限(pjsub -L proc-msgq相当) 指定できる値の範囲は、0～2251799812636672 (2147483647Mi) Byteです。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	size_t
"proc-openfd"	プロセス単位のファイル記述子数制限(pjsub -L proc-openfd相当) 指定できる値の範囲は、0～1048576です。	uint64_t
"proc-psig"	プロセス単位のペンディングシグナル数制限(pjsub -L proc-psig相当) 指定できる値の範囲は、0～2147483647です。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	uint64_t
"proc-filesz"	プロセス単位のファイルサイズ制限(pjsub -L proc-filesz相当) 指定できる値の範囲は、2048～2147483647000000 (2147483647M) Byteです。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	size_t
"proc-stack"	プロセス単位のスタックサイズ制限(pjsub -L proc-stack相当) 指定できる値の範囲は、0～2251799812636672 (2147483647Mi) Byteです。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	size_t
"proc-vmem"	プロセス単位の仮想メモリサイズ制限(pjsub -L proc-vmem相当) 指定できる値の範囲は、0～2251799812636672 (2147483647Mi) Byteです。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	size_t
上記以外の文字列	引数*rscname_p*で示される名前のカスタム資源の資源量(pjsub -L *カスタム資源名* 相当)	char *

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*rscname_p*が不正(NULL)です。

PJCMD_ERROR_INVALID_PARAM

資源量の値が不正です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.1.11 pjcmd_submit_get_job_resource()

```
pjcmd_result_t pjcmd_submit_get_job_resource(const PjcmdHandle_t *handle_p, const char *rscname_p, void *val_p)
```

ハンドルに設定されているジョブに割り当てる資源量を参照します。

[引数]

handle_p

ハンドルへのポインター

rscname_p

値を参照する資源名。指定できる資源名は、関数pjcmd_submit_put_job_resource()と同じです。

val_p

*rscname_p*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*rscname_p*または*val_p*が不正(NULL)です。

PJCMD_ERROR_NODATA

指定した資源については、資源量は設定されていません。

B.1.12 pjcmd_submit_put_mpi_param()

```
pjcmd_result_t pjcmd_submit_put_mpi_param(PjcmdHandle_t *handle_p, pjcmd_submit_mpi_param_t param, const void *val_p)
```

MPIジョブ実行に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するパラメーターの種類(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がchar *型の場合、char *型の値を格納した領域を呼び出し側で用意し、それへのポインター(char **)を*val_p*に指定してください。NULLを指定した場合は、既設定を取り消し、初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_SUBMIT_MPI_SHAPE	プログラム起動時に生成するプロセスの形状(pjsub --mpi shape相当) ・ 計算ノードがFXサーバの場合 ー 1次元形状: "*X*" ー 2次元形状: "*X*x*Y*" ー 3次元形状: "*X*x*Y*x*Z*" ハンドルに資源名"vnode"が設定されている場合は、"1"のみ指定でき、それ以外はエラーになります。 ・ 計算ノードがPRIMERGYサーバの場合 無視されます。	char *
PJCMD_SUBMIT_MPI_PROC	プログラム起動時に生成する最大プロセス数(pjsub --mpi proc相当) 指定できる値の範囲は、1～2147483647です。資源名"shape"で指定した形状が表すノード数×1ノード内のCPUコア数より大きい値を指定すると、ジョブ投入結果がエラーになります。	int
PJCMD_SUBMIT_MPI_MAX_PROC_PER_NODE	プログラムが1ノード内に生成する最大プロセス数(pjsub --mpi max-proc-per-node相当) 指定できる値の範囲は、1～2147483647です。指定した値が、1ノード内のCPUコア数よりも大きい場合、または、PJCMD_SUBMIT_MPI_PROCで指定した値を1ノード内に生成するプロセス数に変換した値のほうが大きい場合は、ジョブ投入結果がエラーになります。 ハンドルに資源名"vnode"が設定されている場合に、本パラメーターを指定した場合は、ジョブ投入結果がエラーになります。	int
PJCMD_SUBMIT_MPI_RANK_MAP_BYNODE	ランク割り付けルール(pjsub --mpi rank-map-bynode相当)	char *

param	**val_p*	**val_p*の型
	・ 計算ノードがFXサーバの場合 **val_p*には "XY"、 "YX"、 "XYZ"、"XZY"、"YXZ"、"YZX"、"ZXY"、または"ZYX"が指定できます。 これらの意味は、pjsub --mpi rank-map-bynode=*rankmap*と同じです。 **val_p*に空文字""を指定すると、pjsub --mpi rank-map-bynodeと同じ意味になります。 ・ 計算ノードがPRIMERGYサーバの場合 **val_p*には空文字を指定してください。pjsub --mpi rank-map-bynodeと同じ意味になります。	
PJCMD_SUBMIT_MPI_RANK_MAP_BYCHIP	ランク割り付けルール(pjsub --mpi rank-map-bychip相当) ・ 計算ノードがFXサーバの場合 **val_p*には "XY"、 "YX"、 "XYZ"、"XZY"、"YXZ"、"YZX"、"ZXY"、または"ZYX"が指定できます。 これらの意味は、pjsub --mpi rank-map-bychip:*rankmap*と同じです。 **val_p*に空文字""を指定すると、pjsub --mpi rank-map-bychipと同じ意味になります。 ・ 計算ノードがPRIMERGYサーバの場合 **val_p*には文字列で整数*n*を指定してください。意味はpjsub --mpi rank-map-bychip=*n*と同じです。	char *
PJCMD_SUBMIT_MPI_RANK_MAP_HOSTFILE	ランクマップホストファイル名(pjsub --mpi rank-map-hostfile相当)	char *
PJCMD_SUBMIT_MPI_ASSIGN_ONLINE_NODE	割り当てノードに、故障ノードが含まれないことを保証するかどうかを指定します(pjsub --mpi assign-online-node相当)。 0: 保証しません(デフォルト)。 1: 保証します。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.1.13 pjcmd_submit_get_mpi_param()

```
pjcmd_result_t pjcmd_submit_get_mpi_param(const PjcmdHandle_t *handle_p, pjcmd_submit_mpi_param_t param, void *val_p)
```

ハンドルに設定されている、MPIジョブ実行に関するパラメーターの値を参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_submit_put_mpi_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.1.14 pjcmd_submit_put_sched_param()

```
pjcmd_result_t pjcmd_submit_put_sched_param(PjcmdHandle_t *handle_p, pjcmd_submit_sched_param_t param, const void *val_p)
```

投入するジョブのスケジューリングに関するパラメーターをハンドル*handle_p*に設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_SUBMIT_SCHED_PRIORITY	ジョブの優先度 (同一ユーザーのジョブにおける優先度、pjsub -p相当) 0から255までの整数が指定できます。未設定時はジョブACL機能の設定に従います。	int
PJCMD_SUBMIT_SCHED_AUTORESTART	ジョブの自動再実行を有効にするか否かを指定します。 0 : 無効にします。 1 : 有効にします。 未設定時は管理者によるジョブ運用管理の設定(papjm.conf、pmpjm.confファイル)に従います。	int
PJCMD_SUBMIT_SCHED_STARTDATE	ジョブの実行開始予定時刻	time_t
PJCMD_SUBMIT_SCHED_VN_POLICY [PG]	仮想ノード配置ポリシー ・ PJCMD_SUBMIT_VN_POLICY_ABSPACK abs-pack (pjsub -P vn-policy=abs-pack相当) ・ PJCMD_SUBMIT_VN_POLICY_PACK pack (pjsub -P vn-policy=pack相当) ・ PJCMD_SUBMIT_VN_POLICY_UNPACK unpack (pjsub -P vn-policy=unpack相当) ・ PJCMD_SUBMIT_VN_POLICY_ABSUNPACK abs-unpack (pjsub -P vn-policy=abs-unpack相当) 未設定時はジョブACL機能の設定に従います。	int
PJCMD_SUBMIT_SCHED_VN_POLICY_N [PG]	仮想ノード配置ポリシーがunpackまたはabs-unpackのときに、物理ノードに配置する仮想ノード数を指定します。 指定できる値の範囲は、1～2147483647です。	int
PJCMD_SUBMIT_SCHED_EXEC_POLICY [PG]	実行モードポリシー ・ PJCMD_SUBMIT_EXEC_POLICY_SIMPLEX simplex (pjsub -P exec-policy=simplex相当) ・ PJCMD_SUBMIT_EXEC_POLICY_SHARE share (pjsub -P exec-policy=share相当) 未設定時はジョブACL機能の設定に従います。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.1.15 pjcmd_submit_get_sched_param()

```
pjcmd_result_t pjcmd_submit_get_sched_param( const PjcmdHandle_t *handle_p, pjcmd_submit_sched_param_t param, void *val_p)
```

ハンドルに設定されているスケジューリングに関するパラメーターの値を参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_submit_put_sched_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.1.16 pjcmd_submit_put_fileio_param()

```
pjcmd_result_t pjcmd_submit_put_fileio_param(PjcmdHandle_t *handle_p, pjcmd_submit_fileio_param_t param, const void *val_p)
```

ジョブ実行時のファイル入出力に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がchar *型の場合、char *型の値を格納した領域を呼び出し側で用意し、それへのポインター(char **)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_SUBMIT_FILEIO_OFILE	ジョブの標準出力の出力先パス(pjsub -o|--out相当) 未設定時は、ファイル"*ジョブ名.ジョブID*.out"になります。	char *
PJCMD_SUBMIT_FILEIO_EFILE	ジョブの標準エラー出力の出力先パス(pjsub -e|--err相当) 未設定時は、ファイル"*ジョブ名.ジョブID*.err"になります。	char *
PJCMD_SUBMIT_FILEIO_SFILE	ジョブの統計情報ファイルの出力先パス(pjsub --spath相当) 未設定時は、"*ジョブ名 ジョブID*.stats"になります。	char *
PJCMD_SUBMIT_FILEIO_SFILE_MODE	ジョブ統計情報ファイルの出力方法 ・ PJCMD_SUBMIT_SFILE_MODE_DISABLE ジョブ統計情報ファイルは出力しません(デフォルト)。 ・ PJCMD_SUBMIT_SFILE_MODE_JOB ジョブ統計情報ファイルにジョブ情報だけ出力します(pjsub -s|--stats相当)。 ・ PJCMD_SUBMIT_SFILE_MODE_JOB_AND_NODE ジョブ統計情報ファイルにジョブ情報をノード情報の両方を出力します(pjsub -S|--STATS相当)。	int
PJCMD_SUBMIT_FILEIO_MERGE	ジョブの標準エラー出力を標準出力と同じファイルに出力するか否かを指定します。	int

param	**val_p*	**val_p*の型
	0 : 同じファイルに出力しません(デフォルト)。 1 : 同じファイルに出力します。	

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.1.17 pjcmd_submit_get_fileio_param()

```
pjcmd_result_t pjcmd_submit_get_fileio_param(const PjcmdHandle_t *handle_p, pjcmd_submit_fileio_param_t param, void *val_p)
```

ハンドルに設定されているジョブ実行時のファイル入出力に関するパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数 pjcmd_submit_put_fileio_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.1.18 pjcmd_submit_put_llio_param()

```
pjcmd_result_t pjcmd_submit_put_llio_param(PjcmdHandle_t *handle_p, pjcmd_submit_llio_param_t param, const void *val_p)
```

ジョブ実行時の階層化ストレージの利用に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がsize_t型の場合、size_t型の値を格納した領域を呼び出し側で用意し、それへのポインター(size_t *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_SUBMIT_LLIO_SHAREDTMP_SIZE	第1階層ストレージ上の共有テンポラリ領域のサイズ 指定できる値の範囲は0～2251799812636672 (2147483647Mi) Byteで、1048576 (1Mi) Byte単位で指定してください。	size_t
PJCMD_SUBMIT_LLIO_LOCALTMP_SIZE	第1階層ストレージ上のノード内テンポラリ領域のサイズ 1つのジョブが利用できる共有テンポラリ領域のサイズは、指定した値×ジョブに割り当てられる計算ノード数です。 指定できる値の範囲は0～2251799812636672 (2147483647Mi) Byteで、1048576 (1Mi) Byte単位で指定してください。	size_t
PJCMD_SUBMIT_LLIO_AUTO_READAHEAD	ジョブが複数回続けて階層化ストレージ内の連続した領域を読み込もうとした場合、自動的に先読みをするか否かの動作	int

param	**val_p*	**val_pの型*
	0: 先読みをしません。 1: 先読みをします。	
PJCMD_SUBMIT_LLIO_ASYNC_CLOSE	階層化ストレージ上のファイルのクローズを非同期クローズにするか否かの動作 0: 同期クローズ 1: 非同期クローズ	int
PJCMD_SUBMIT_LLIO_CN_CACHED_WRITE_SIZE	第1階層ストレージへの書き込み時にキャッシュするか否かのしきい値 第1階層ストレージへの書き込みの際に、書出しサイズが指定した値以下の場合は、すぐにはストレージに書き出さず、一時的に計算ノード内キャッシュに蓄えます。 指定できる値の範囲は0～2251799812636672 (2147483647Mi) Byteで、4194304 (4Mi) Byte単位で指定してください。	size_t
PJCMD_SUBMIT_LLIO_CN_CACHE_SIZE	計算ノード内キャッシュのサイズ 指定できる値の範囲は4194304 (4Mi)～2251799812636672 (2147483647Mi) Byteです。 また、値はストライプサイズ(パラメーターPJCMD_SUBMIT_LLIO_STRIPE_SIZEの値)×10以上でなければ、ジョブ投入依頼がエラーになります。	size_t
PJCMD_SUBMIT_LLIO_CN_READ_CACHE	ジョブが階層化ストレージからファイルを読むときに、計算ノード内にキャッシュするか否かの動作 0: キャッシュしません。 1: キャッシュします。	int
PJCMD_SUBMIT_LLIO_SIO_READ_CACHE	データの読み込み時に、第2階層ストレージの領域から読み込んだデータを第1階層ストレージにキャッシュするか否か。 0: キャッシュしません。 1: キャッシュします。	int
PJCMD_SUBMIT_LLIO_STRIPE_COUNT	第1階層ストレージにファイルを分散配置する際のファイル当たりのストライプ数 指定できる値の範囲は1～2147483647です。	int
PJCMD_SUBMIT_LLIO_STRIPE_SIZE	第1階層ストレージにファイルを分散配置する際のストライプサイズ 指定できる値の範囲は65536(64Ki)～4294901760(4194240Ki)Byteで、65536(64Ki)Byte単位で指定してください。	size_t
PJCMD_SUBMIT_LLIO_UNCOMPLETED_FILEINFO_PATH	未書出しファイル一覧の出力先パス ジョブ終了時に、キャッシュから第2階層ストレージへの書出しが完了しなかったファイルの情報の出力先パスです。	char *
PJCMD_SUBMIT_LLIO_PERF	LLIO性能情報を出力するか否かの動作	int

param	**val_p*	**val_p*の型
	0: 出力しません。 1: 出力します。	
PJCMD_SUBMIT_LLIO_PERF_PATH	LLIO性能情報の出力先を変更したい場合に、パスを指定します。	char *

上記パラメーターを設定しない場合は、ジョブACL機能の設定に従います。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.1.19 pjcmd_submit_get_llio_param()

```
pjcmd_result_t pjcmd_submit_get_llio_param(const PjcmdHandle_t *handle_p, pjcmd_submit_llio_param_t param, void *val_p)
```

ハンドルに設定されているジョブ実行時の階層化ストレージの利用に関するパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_submit_put_llio_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.1.20 pjcmd_submit_create_scriptfile_from_stdin()

```
char *pjcmd_submit_create_scriptfile_from_stdin(const char *basedir_p, const char *filename_p)
```

標準入力の内容をジョブスクリプトとして作成します。コマンドライン引数にジョブスクリプトが指定されない場合に、標準入力からジョブの内容を取得し、関数pjcmd_submit_put_param()で、ジョブスクリプトとして指定するために使います。

[引数]

basedir_p

ジョブスクリプトを作成するディレクトリのパス。
関数を呼び出すユーザーに対するアクセス権が必要です。NULLを指定した場合は、関数を呼び出したときのカレントディレクトリになります。

filename_p

ディレクトリ*basedir_p*配下に作成するジョブスクリプトのファイル名。NULLを指定した場合は、既存のファイルと重複しないファイル名を自動的に決定します。

[返り値]

標準入力の内容が格納されたジョブスクリプトのパス名。この領域は呼び出し側が解放する必要があります。失敗時はNULLを返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_OPEN

ジョブスクリプトの作成に失敗しました(権限がない、同名のファイルが存在する、またはパス名が不適切)、または入力が中断されました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

[注意]

ー 作成したジョブスクリプトファイルは、ジョブ投入後、利用者側の責任で削除する必要があります。

ー 本関数は、標準入力が閉じられるまで復帰しません。

B.1.21 pjcmd_submit_create_scriptfile_by_args()

```
char *pjcmd_submit_create_scriptfile_by_args(const char *basedir_p, const char *filename_p, int argc, const char *argv_p[])
```

コマンドライン引数を内容とするジョブスクリプトを作成します。

[引数]

basedir_p

ジョブスクリプトを作成するディレクトリ。
関数を呼び出すユーザーに対するアクセス権が必要です。NULLを指定した場合は、関数を呼び出したときのカレントディレクトリになります。

filename_p

ディレクトリ*basedir_p*配下に作成するジョブスクリプトのファイル名。NULLを指定した場合は、既存のファイルと重複しないファイル名を自動的に決定します。

argc

コマンドライン引数の数

argv_p[]

コマンドライン引数の配列

[返り値]

コマンドライン引数の内容が格納されたジョブスクリプトのパス。失敗時はNULLを返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*argc*が0または*argv_p*がNULLです。

PJCMD_ERROR_OPEN

ジョブスクリプトの作成に失敗しました(権限がない、同名のファイルが存在する、またはパスが不適切)。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

[注意]

作成したジョブスクリプトファイルは、利用者側の責任で削除する必要があります。

B.1.22 pjcmd_submit_set_callback()

```
pjcmd_result_t pjcmd_submit_set_callback(
    PjcmdHandle_t *handle_p,
    void (*job_accept_callback_func_p)(const PjcmdSubjobid_t *),
    int (*start_wait_callback_func_p)(const PjcmdSubjobid_t *),
    void (*job_start_callback_func_p)(const PjcmdSubjobid_t *),
    void (*job_end_callback_func_p)(const PjcmdSubjobid_t *))
```

会話型ジョブの処理の進行に合わせて、特定のタイミングで呼び出されるコールバック関数を登録します。利用者が特定のタイミングで独自の処理を呼び出したい場合に利用します。例えば、会話型ジョブの処理の経過を示すメッセージを出力するために利用できます。
コールバック関数としてNULLポインターが指定された場合は、そのコールバック関数は設定されていないとみなします。

[引数]

handle_p

ハンドルへのポインター

job_accept_callback_func_p

会話型ジョブが受け付けられたときに呼び出す関数へのポインター

start_wait_callback_func_p

会話型ジョブが実行待ちになったときに呼び出す関数へのポインター。この関数を登録すると、ジョブの実行が始まるまで3秒間隔で呼び出されます。

job_start_callback_func_p

会話型ジョブの実行開始時に呼び出す関数へのポインター

job_end_callback_func_p

会話型ジョブが終了したときに呼び出す関数へのポインター

コールバック関数には、呼び出し時に実行する会話型ジョブのジョブIDがサブジョブID構造体で渡されます。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

B.1.23 pjcmd_submit_execute()

```
PjcmdResp_t *pjcmd_submit_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、ジョブの投入をジョブ運用管理機能に依頼します。この関数は、ログインノードおよび計算クラスタ管理ノードで呼び出せます。

[引数]

handle_p

ハンドルへのポインター

[返り値]

ジョブ投入依頼の応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。ジョブ投入依頼が失敗した場合は、NULLを返し、原因がpjcmd_errcodeに設定されます。
なお、応答情報は依頼が成功したか否かを示します。ジョブの投入がジョブ運用管理機能に受け付けられたか否かは、関数pjcmd_get_jobresult_info()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノードと計算クラスタ管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_NOENT

ハンドルに設定されているファイルまたはディレクトリが存在しません。

PJCMD_ERROR_ACCESS

ハンドルに設定されているファイルへのアクセス権がありません。

PJCMD_ERROR_OPEN

ハンドルに設定されているファイルのオープンに失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

B.1.24 pjcmd_submit_executev()

```
PjcmdResp_t *pjcmd_submit_executev(const PjcmdHandle_t **handle_pp, int n)
```

ハンドルを配列で指定する以外は、関数pjcmd_submit_execute()と同じです。ただし、2つ以上のハンドルを指定する場合は、すべてステップジョブのサブジョブの投入に関するハンドルでなければいけません。この関数は、ログインノードおよび計算クラスタ管理ノードで呼び出せます。

[引数]

handle_pp

ハンドルへのポインターの配列

n

ハンドルの個数

[返り値]

ジョブ投入依頼の応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。ジョブ投入依頼が失敗した場合は、NULLを返し、原因がpjcmd_errcodeに設定されます。
なお、応答情報は依頼が成功したか否かを示します。ジョブの投入がジョブ運用管理機能に受け付けられたか否かは、関数pjcmd_get_jobresult_info()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入用のハンドルではありません。
- ハンドルが複数指定されている場合に、ステップジョブ以外のジョブのハンドルが含まれています。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノードと計算クラスタ管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_NOENT

ハンドルに設定されているファイルまたはディレクトリが存在しません。

PJCMD_ERROR_ACCESS

ハンドルに設定されているファイルへのアクセス権がありません。

PJCMD_ERROR_OPEN

ハンドルに設定されているファイルのオープンに失敗しました。

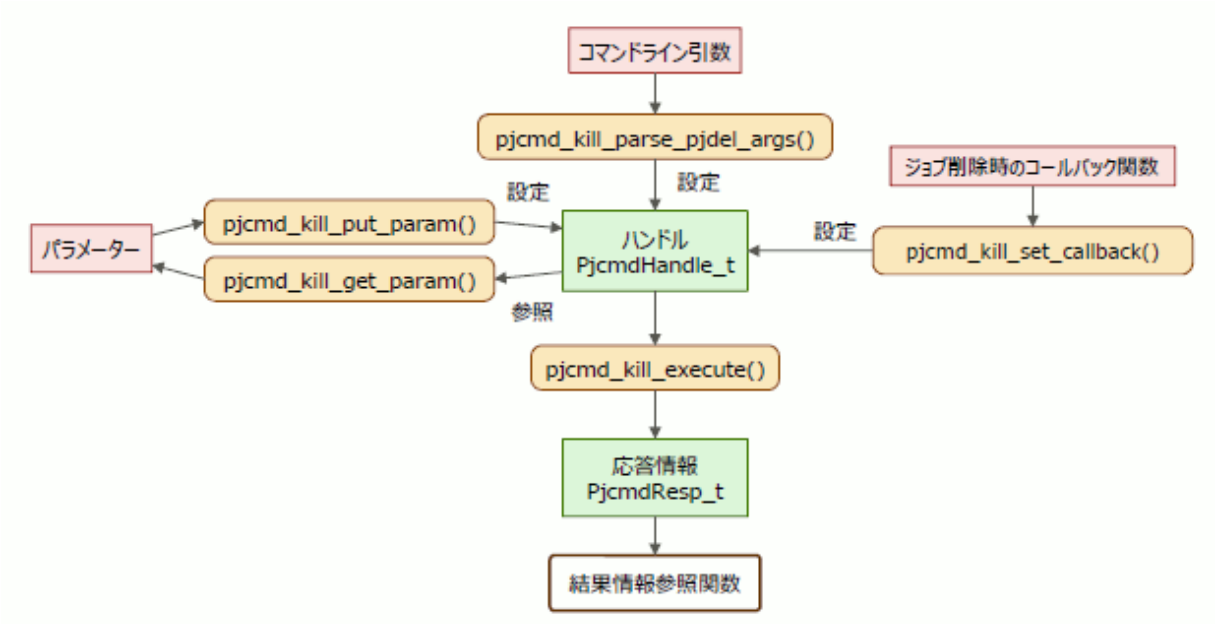
PJCMD_ERROR_INTERNAL

内部エラー

B.2 ジョブの削除

ここでは、ジョブを削除(キャンセル)するための関数を説明します。

図B.3 ジョブの削除依頼



B.2.1 pjcmd_kill_parse_pjdel_args()

```
pjcmd_result_t pjcmd_kill_parse_pjdel_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjdelコマンドのオプション仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ削除用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションでない引数が配列*argv_pp*[]の後ろに移動します。
正常に終了した場合は、変数pjcmd_optindがジョブID(オプション以外の最初の引数)を指します。ジョブIDについては呼び出し側で、ハンドルに設定する必要があります。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1]がそのオプションを指します。

B.2.2 pjcmd_kill_put_param()

```
pjcmd_result_t pjcmd_kill_put_param(PjcmdHandle_t *handle_p, pjcmd_kill_param_t param, const void *val_p)
```

ジョブ削除に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_KILL_ENFORCE	プロローグ、エピローグスクリプトを実行中の場合に中断させるか否かの指定(pjdel --enforce相当)。 0 : 中断しません(デフォルト)。 1 : 強制中断し、ジョブを削除します。	int
PJCMD_KILL_REASON	ジョブ統計情報に削除理由として出力するメッセージ(pjdel --reasonに相当)。 文字列は、終端のNULL文字を含め64バイト以内でなければいけません。使用できる文字は半角英数字と表示可能な記号です。	char *
PJCMD_KILL_NO_STATS	削除するジョブがQUEUED状態の場合、そのジョブのジョブ統計情報ファイル(.statsファイル)の出力を抑止する指定(pjdel --no-stats相当)。 0: 抑止しません(デフォルト)。 1: 抑止します。	int
PJCMD_KILL_NO_HISTORY	削除するジョブがQUEUED状態の場合、pjstatコマンドの-Hオプションで出力されるジョブの履歴情報に、そのジョブの情報を出力することを抑止する指定(pjdel --no-history相当)。 0: 抑止しません(デフォルト)。 1: 抑止します。	int
PJCMD_KILL_LLIO_FLUSH	ジョブを削除する前に、ジョブの実行時間の制限値を超えない範囲で、未書き出しファイルのフラッシュ完了待ちをするか否かの指定(pjdel --llio-flush相当)。 0 : フラッシュ完了を待ち合わせません(デフォルト)。 1 : フラッシュ完了を待ち合わせます。	int
PJCMD_KILL_HELP	pjdelコマンドの--helpオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、ジョブの削除操作には影響しません。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ削除用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.2.3 pjcmd_kill_get_param()

```
pjcmd_result_t pjcmd_kill_get_param(const PjcmdHandle_t *handle_p, pjcmd_kill_param_t param, void *val_p)
```

ジョブの削除用のハンドルに設定されているパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_kill_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ削除用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.2.4 pjcmd_kill_set_callback()

```
pjcmd_result_t pjcmd_kill_set_callback(
    PjcmdHandle_t *handle_p,
    void (*kill_wait_callback_func_p)(void),
    void (*kill_accept_callback_func_p)(void))
```

ジョブの削除依頼の進行に合わせて、特定のタイミングで呼び出されるコールバック関数を登録します。利用者が特定のタイミングで独自の処理を呼び出したい場合に利用します。例えば、ジョブの削除依頼の受付け待ちを示すメッセージを出力するために利用できます。コールバック関数としてNULLポインターが指定された場合は、そのコールバック関数は設定されていないとみなします。

[引数]

handle_p

ハンドルへのポインター

kill_wait_callback_func_p

ジョブの削除依頼の受付け待ちになったときに呼び出す関数へのポインター。この関数を登録すると、ジョブの削除依頼の受付けが完了するまで3秒間隔で呼び出されます。

kill_accept_callback_func_p

ジョブの削除依頼の受付けが完了したときに呼び出す関数へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ削除用のハンドルではありません。

B.2.5 pjcmd_kill_execute()

```
PjcmdResp_t *pjcmd_kill_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、ジョブの削除をジョブ運用管理機能に依頼します。この関数は、ログインノードおよび計算クラスタ管理ノードで呼び出せます。

[引数]

handle_p

ハンドルへのポインター

[返り値]

ジョブの削除の応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。ジョブの削除依頼が失敗した場合は、NULLを返し、原因がpjcmd_errcodeに設定されます。
なお、応答情報は依頼が成功したか否かを示します。ジョブの削除がジョブ運用管理機能に受け付けられたか否かは、関数pjcmd_get_jobresult_info()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ削除用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノードと計算クラスタ管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

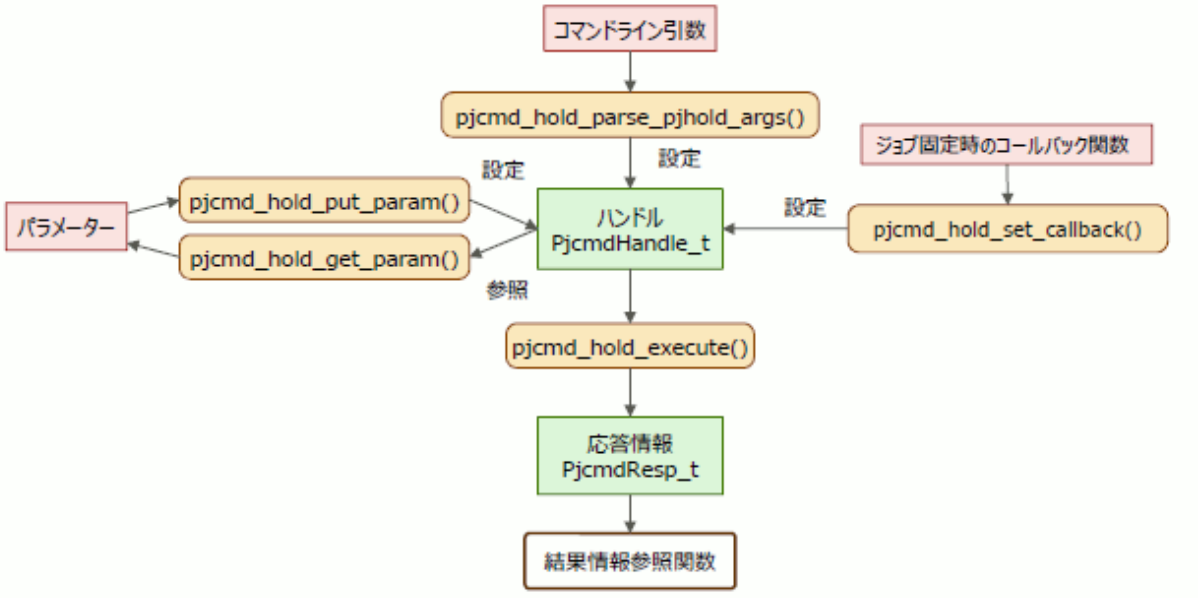
PJCMD_ERROR_INTERNAL

内部エラー

B.3 ジョブの固定

ここでは、ジョブを固定するための関数を説明します。

図B.4 ジョブの固定依頼



B.3.1 pjcmd_hold_parse_pjhold_args()

```
pjcmd_result_t pjcmd_hold_parse_pjhold_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjholdコマンドのオプション仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ固定用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションでない引数が*argv_pp*[]の後ろに移動します。
正常に終了した場合は、変数pjcmd_optindがジョブID(オプション以外の最初の引数)を指します。ジョブIDについては呼び出し側で、ハンドルに設定する必要があります。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1]がそのオプションを指します。

B.3.2 pjcmd_hold_put_param()

```
pjcmd_result_t pjcmd_hold_put_param(PjcmdHandle_t *handle_p, pjcmd_hold_param_t param, const void *val_p)
```

ジョブの固定に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_HOLD_ENFORCE	プロローグ、エピローグスクリプトを実行中の場合に中断させるか否かの指定。 0 : 中断しません(デフォルト)。 1 : 強制中断し、ジョブを固定します。	int
PJCMD_HOLD_REASON	ジョブ統計情報に固定理由として出力するメッセージ(pjhold --reasonに相当)。 文字列は、終端のNULL文字を含め64バイト以内でなければいけません。使用できる文字は半角英数字と表示可能な記号です。	char *
PJCMD_HOLD_LLIO_FLUSH	ジョブを固定する前に、ジョブの実行時間の制限値を超えない範囲で、未書き出しファイルのフラッシュ完了待ちをするか否かの指定(pjhold --llio-flushに相当)。 0 : フラッシュ完了を待ち合わせません(デフォルト)。 1 : フラッシュ完了を待ち合わせます。	int
PJCMD_HOLD_HELP	pjholdコマンドの--helpオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、ジョブの固定操作には影響しません。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの固定用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.3.3 pjcmd_hold_get_param()

```
pjcmd_result_t pjcmd_hold_get_param(const PjcmdHandle_t *handle_p, pjcmd_hold_param_t param, void *val_p)
```

ジョブの固定用のハンドルに設定されているパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_hold_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの固定用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.3.4 pjcmd_hold_set_callback()

```
pjcmd_result_t pjcmd_hold_set_callback(
    PjcmdHandle_t *handle_p,
    void (*hold_wait_callback_func_p)(void),
    void (*hold_accept_callback_func_p)(void))
```

ジョブの固定依頼の進行に合わせて、特定のタイミングで呼び出されるコールバック関数を登録します。利用者が特定のタイミングで独自の処理を呼び出したい場合に利用します。例えば、ジョブの固定依頼の受付け待ちを示すメッセージを出力するために利用できます。コールバック関数としてNULLポインターが指定された場合は、そのコールバック関数は設定されていないとみなします。

[引数]

handle_p

ハンドルへのポインター

hold_wait_callback_func_p

ジョブの固定依頼の受付け待ちになったときに呼び出す関数へのポインター。この関数を登録すると、ジョブの固定依頼の受付けが完了するまで3秒間隔で呼び出されます。

hold_accept_callback_func_p

ジョブの固定依頼の受付けが完了したときに呼び出す関数へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ固定用のハンドルではありません。

B.3.5 pjcmd_hold_execute()

```
PjcmdResp_t *pjcmd_hold_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、ジョブの固定をジョブ運用管理機能に依頼します。この関数は、ログインノードおよび計算クラスタ管理ノードで呼び出せます。

[引数]

handle_p

ハンドルへのポインター

[返り値]

ジョブ固定の応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。ジョブの固定依頼が失敗した場合は、NULLを返し、原因がpjcmd_errcodeに設定されます。
なお、応答情報は、依頼が成功したか否かを示します。ジョブの固定がジョブ運用管理機能に受け付けられたか否かは、関数pjcmd_get_jobresult_info()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ固定用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノードと計算クラスタ管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

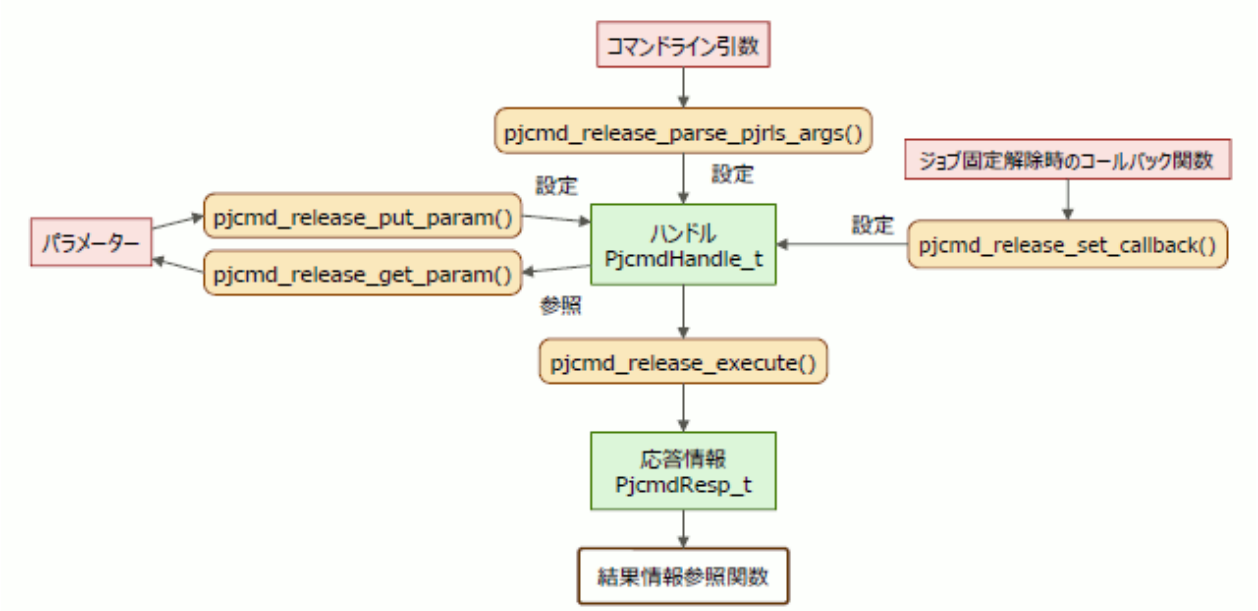
PJCMD_ERROR_INTERNAL

内部エラー

B.4 ジョブの固定解除

ここでは、ジョブの固定を解除するための関数を説明します。

図B.5 ジョブの固定解除依頼



B.4.1 pjcmd_release_parse_pjrls_args()

```
pjcmd_result_t pjcmd_release_parse_pjrls_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjrlsコマンドのオプション仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの固定解除用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションでない引数が*argv_p*[]の後ろに移動します。
正常に終了した場合は、変数pjcmd_optindがジョブID(オプション以外の最初の引数)を指します。ジョブIDについては呼び出し側で、ハンドルに設定する必要があります。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1]がそのオプションを指します。

B.4.2 pjcmd_release_put_param()

```
pjcmd_result_t pjcmd_release_put_param(PjcmdHandle_t *handle_p, pjcmd_release_param_t param, const void *val_p)
```

ジョブの固定解除に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_RELEASE_HELP	pjrlsコマンドの--helpオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、ジョブの固定解除操作には影響しません。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの固定解除用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.4.3 pjcmd_release_get_param()

```
pjcmd_result_t pjcmd_release_get_param(const PjcmdHandle_t *handle_p, pjcmd_release_param_t param, void *val_p)
```

ジョブの固定解除用のハンドルに設定されているパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_release_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの固定解除用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.4.4 pjcmd_release_set_callback()

```
pjcmd_result_t pjcmd_release_set_callback(
    PjcmdHandle_t *handle_p,
    void (*release_wait_callback_func_p)(void),
    void (*release_accept_callback_func_p)(void))
```

ジョブの固定解除依頼の進行に合わせて、特定のタイミングで呼び出されるコールバック関数を登録します。利用者が特定のタイミングで独自の処理を呼び出したい場合に利用します。例えば、ジョブの固定解除依頼の受付け待ちを示すメッセージを出力するために利用できます。
コールバック関数としてNULLポインターが指定された場合は、そのコールバック関数は設定されていないとみなします。

[引数]

handle_p

ハンドルへのポインター

release_wait_callback_func_p

ジョブの固定解除依頼の受付け待ちになったときに呼び出す関数へのポインター。この関数を登録すると、ジョブの固定解除依頼の受付けが完了するまで3秒間隔で呼び出されます。

release_accept_callback_func_p

ジョブの固定解除依頼の受付けが完了したときに呼び出す関数へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ固定解除用のハンドルではありません。

B.4.5 pjcmd_release_execute()

```
PjcmdResp_t *pjcmd_release_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、ジョブの固定解除をジョブ運用管理機能に依頼します。この関数は、ログインノードおよび計算クラスタ管理ノードで呼び出せます。

[引数]

handle_p

ハンドルへのポインター

[返り値]

ジョブ固定解除の応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。ジョブの固定解除の依頼が失敗した場合は、NULLを返し、原因がpjcmd_errcodeに設定されます。
なお、応答情報は依頼が成功したか否かを示します。ジョブの固定解除がジョブ運用管理機能に受け付けられたか否かは、関数pjcmd_get_jobresult_info()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの固定解除用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノードと計算クラスタ管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

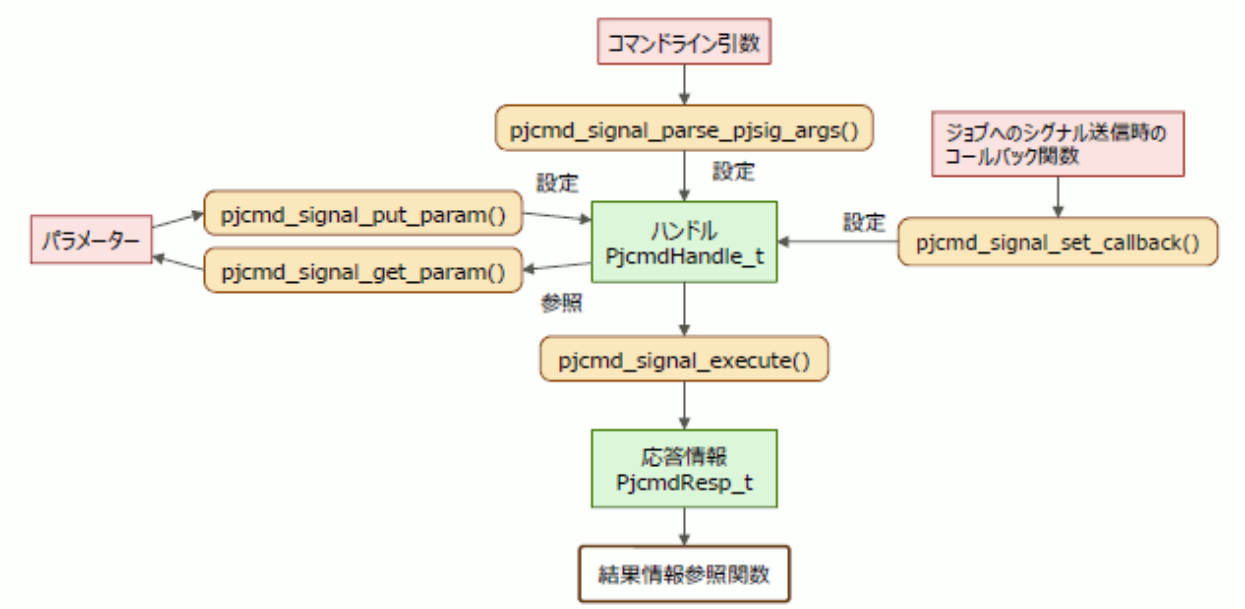
PJCMD_ERROR_INTERNAL

内部エラー

B.5 ジョブへのシグナル送信

ここでは、ジョブにシグナルを送信するための関数を説明します。

図B.6 ジョブへのシグナル送信依頼



B.5.1 pjcmd_signal_parse_pjsig_args()

```
pjcmd_result_t pjcmd_signal_parse_pjsig_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjsigコマンドのオプション仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- シグナル送信用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションでない引数が*argv_pp*[]の後ろに移動します。
正常に終了した場合は、変数pjcmd_optindがジョブID(オプション以外の最初の引数)を指します。ジョブIDについては呼び出し側で、ハンドルに設定する必要があります。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1]がそのオプションを指します。

B.5.2 pjcmd_signal_put_param()

```
pjcmd_result_t pjcmd_signal_put_param(PjcmdHandle_t *handle_p, pjcmd_signal_param_t param, const void *val_p)
```

ジョブへのシグナル送信に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_SIGNAL_SIGNUM	送信するシグナル番号。 1から64までの整数が指定できます。	int
PJCMD_SIGNAL_SIGNAME	送信するシグナル名。 シグナル名は、送信先の計算ノードで認識できる名称(*)であり、15文字以内(終端のNULL文字は除く)でなければいけません。 (*)具体的にはヘッダーファイルsignal.hやmanページsignal(7)で示されている名称で、"SIGHUP"や"SIGKILL"などです。	char *
PJCMD_SIGNAL_HELP	pjsigコマンドの--helpオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、シグナル送信操作には影響しません。	int

パラメーターPJCMD_SIGNAL_SIGNUMおよびPJCMD_SIGNAL_SIGNAMEは、最後に設定されたほうが有効になります。どちらもハンドルに設定されていない場合、関数pjcmd_signal_execute()呼び出し時にエラーになります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブへのシグナル送信用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

[注意]

シグナル番号は、送信先ノード(ジョブが実行されている計算ノード)のOSの仕様に従ってください。

B.5.3 pjcmd_signal_get_param()

```
pjcmd_result_t pjcmd_signal_get_param(const PjcmdHandle_t *handle_p, pjcmd_signal_param_t param, void *val_p)
```

シグナル送信用のハンドルに設定されているパラメーターの値を参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_signal_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- シグナル送信用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.5.4 pjcmd_signal_set_callback()

```
pjcmd_result_t pjcmd_signal_set_callback(
    PjcmdHandle_t *handle_p,
    void (*signal_wait_callback_func_p)(void),
    void (*signal_accept_callback_func_p)(void))
```

ジョブへのシグナル送信依頼の進行に合わせて、特定のタイミングで呼び出されるコールバック関数を登録します。利用者が特定のタイミングで独自の処理を呼び出したい場合に利用します。例えば、ジョブへのシグナル送信依頼の受付け待ちを示すメッセージを出力するために利用できます。
コールバック関数としてNULLポインターが指定された場合は、そのコールバック関数は設定されていないとみなします。

[引数]

handle_p

ハンドルへのポインター

signal_wait_callback_func_p

ジョブへのシグナル送信依頼の受付け待ちになったときに呼び出す関数へのポインター。この関数を登録すると、ジョブへのシグナル送信依頼の受付けが完了するまで3秒間隔で呼び出されます。

signal_accept_callback_func_p

ジョブへのシグナル送信依頼の受付けが完了したときに呼び出す関数へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブへのシグナル送信用のハンドルではありません。

B.5.5 pjcmd_signal_execute()

```
PjcmdResp_t *pjcmd_signal_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、ジョブへのシグナル送信をジョブ運用管理機能に依頼します。この関数は、ログインノードおよび計算クラスタ管理ノードで呼び出せます。

[引数]

handle_p

ハンドルへのポインター

[返り値]

シグナル送信の応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。ジョブへのシグナル送信の依頼に失敗した場合は、NULLを返し、原因がpjcmd_errcodeに設定されます。
なお、応答情報は依頼が成功したか否かを示します。ジョブへのシグナル送信の依頼がジョブ運用管理機能に受け付けられたか否かは、関数pjcmd_get_jobresult_info()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブへのシグナル送信用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノードと計算クラスタ管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_INTERNAL

内部エラー

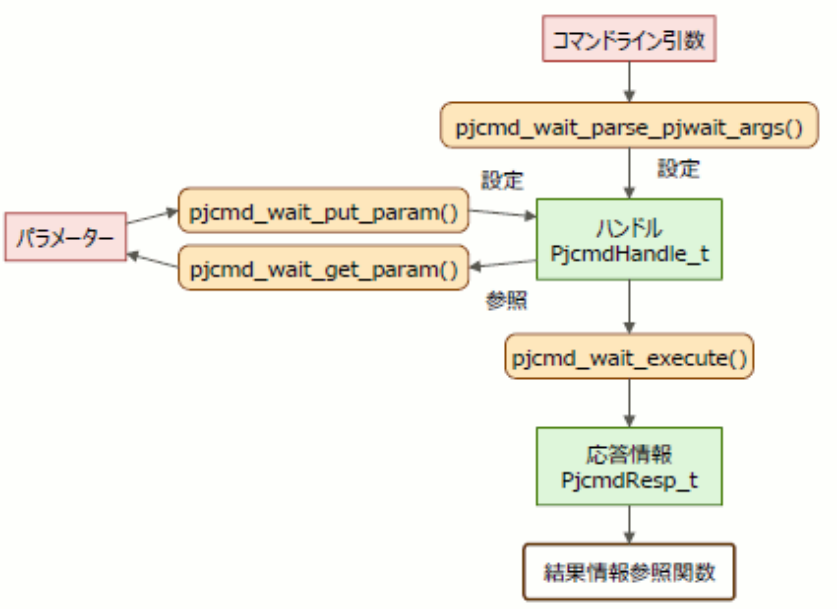
[注意]

このシグナル送信依頼関数は送信の依頼をするだけなので、送信先ノードで認識できないシグナル番号を指定してもエラーにはなりません。正しく送信されたかどうかは、利用者側で確認する必要があります。

B.6 ジョブの終了待ち合わせ

ここでは、ジョブの終了を待ち合わせるための関数を説明します。

図B.7 ジョブの終了待ち合わせ依頼



B.6.1 pjcmd_wait_parse_pjwait_args()

```
pjcmd_result_t pjcmd_wait_parse_pjwait_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjwaitコマンドのオプション仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの終了待ち合わせ用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションでない引数が*argv_pp*[]の後ろに移動します。
正常に終了した場合は、変数pjcmd_optindがジョブID(オプション以外の最初の引数)を指します。ジョブIDについては呼び出し側で、ハンドルに設定する必要があります。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1]がそのオプションを指します。

B.6.2 pjcmd_wait_put_param()

```
pjcmd_result_t pjcmd_wait_put_param(PjcmdHandle_t *handle_p, pjcmd_wait_param_t param, const void *val_p)
```

ジョブの終了待ち合わせに関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_WAIT_MODE	ジョブの終了を待ち合わせるモード、すなわち関数pjcmd_wait_execute()の復帰条件を指定します(pjwait -w相当)。 ・ PJCMD_WAIT_ALL すべてのジョブが終了したら復帰します(デフォルト)。 ・ PJCMD_WAIT_ANY 1つでもジョブが終了したら復帰します。 ・ PJCMD_WAIT_NONE ジョブの終了を待ち合わせず、即復帰します。	int
PJCMD_WAIT_Z	pjwaitコマンドの-zオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、ジョブの終了待ち合わせ操作には影響しません。	int
PJCMD_WAIT_HELP	pjwaitコマンドの--helpオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、ジョブの終了待ち合わせ操作には影響しません。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの終了待ち合わせ用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.6.3 pjcmd_wait_get_param()

```
pjcmd_result_t pjcmd_wait_get_param(const PjcmdHandle_t *handle_p, pjcmd_wait_param_t param, void *val_p)
```

ジョブの終了待ち合わせ用のハンドルに設定されているパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_wait_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの終了待ち合わせ用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.6.4 pjcmd_wait_execute()

```
PjcmdResp_t *pjcmd_wait_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、ジョブの終了待ち合わせをジョブ運用管理機能に依頼します。この関数は、ログインノードおよび計算クラスタ管理ノードで呼び出せます。

[引数]

handle_p

ハンドルへのポインター

[返り値]

ジョブの終了待ち合わせの応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。ジョブの終了待ち合わせの依頼が失敗した場合は、NULLを返し、原因がpjcmd_errcodeに設定されます。
なお、応答情報は依頼が成功したか否かを示します。ジョブの終了待ち合わせの結果は、関数pjcmd_get_jobresult_info()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。

- ジョブの終了待ち合わせ用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノードと計算クラスタ管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

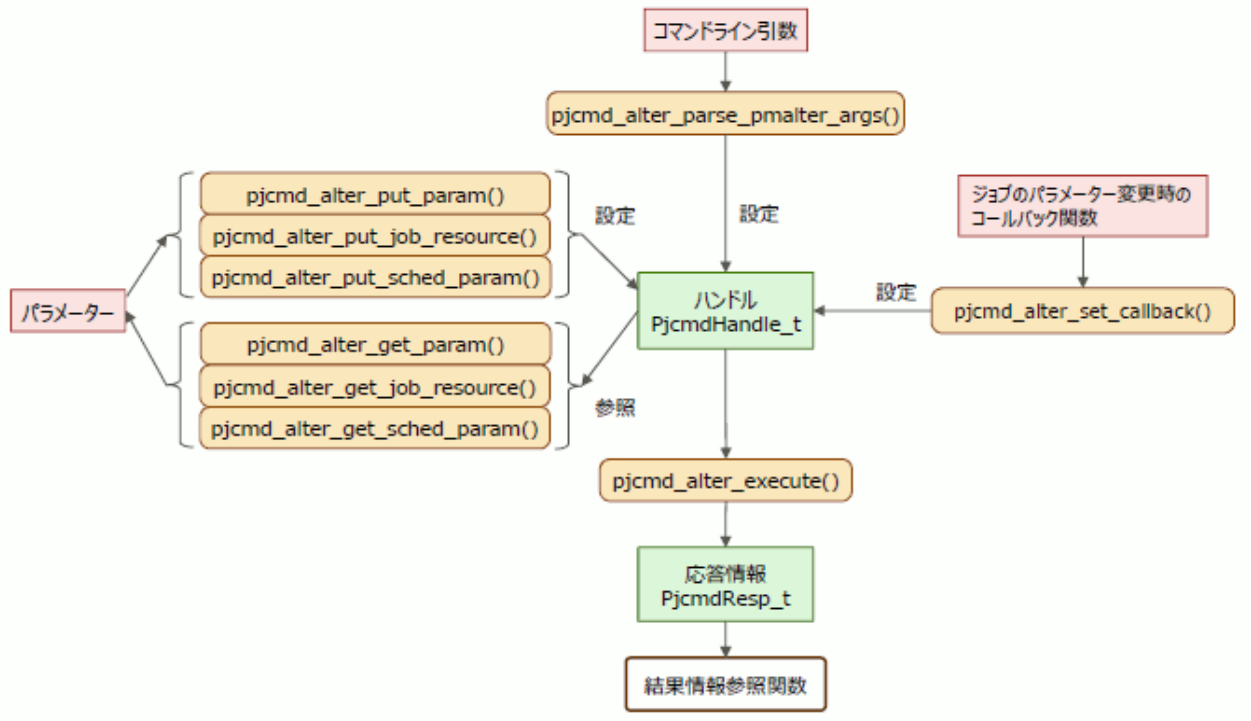
PJCMD_ERROR_INTERNAL

内部エラー

B.7 ジョブのパラメーター変更

ここでは、ジョブのパラメーターを変更するための関数を説明します。

図B.8 ジョブのパラメーター変更依頼



B.7.1 pjcmd_alter_parse_pmalter_args()

```
pjcmd_result_t pjcmd_alter_parse_pmalter_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjalterおよびpmalterコマンドのオプション仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブのパラメーター変更用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションでない引数が*argv_pp*[]の後ろに移動します。
正常に終了した場合は、変数pjcmd_optindがジョブID(オプション以外の最初の引数)を指します。ジョブIDについては呼び出し側で、ハンドルに設定する必要があります。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1]がそのオプションを指します。

[注意]

本関数では、pjalterコマンドとpmalterコマンド両方のオプションについてハンドルに設定できます。しかし、管理者向けのコマンドpmalterコマンドだけに指定できる-Pオプションがハンドルに設定されている場合、ジョブ運用管理機能への依頼(関数pmcmd_alter_execute())をするには、管理者権限が必要です。

B.7.2 pjcmd_alter_put_param()

```
pjcmd_result_t pjcmd_alter_put_param(PjcmdHandle_t *handle_p, pjcmd_alter_param_t param, const void *val_p)
```

変更するジョブのパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がchar *型の場合、char *型の値を格納した領域を呼び出し側で用意し、それへのポインター(char **)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_ALTER_CLUSTER	クラスタ名(1つだけ。pmalter -c相当)。 設定をしない場合は、環境変数PXMYCLSTの値が適用されます。それも設定されていない場合は、関数pjcmd_alter_execute()がエラーになります。 このパラメーターは、システム管理ノードで呼び出された場合だけ有効になります。システム管理ノード以外では、設定は無視され、呼び出しノードが属するクラスタ名が適用されます。	char *
PJCMD_ALTER_HELP	pjalter、pmalterコマンドの--helpオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、ジョブのパラメーター変更には影響しません。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブのパラメーター変更用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.7.3 pjcmd_alter_get_param()

```
pjcmd_result_t pjcmd_alter_get_param(const PjcmdHandle_t *handle_p, pjcmd_alter_param_t param, void *val_p)
```

ジョブパラメーター変更用のハンドルに設定されている内容を参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_alter_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブのパラメーター変更用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.7.4 pjcmd_alter_put_job_resource()

```
pjcmd_result_t pjcmd_alter_put_job_resource(PjcmdHandle_t *handle_p, char *rscname_p, const void *val_p)
```

変更したいジョブの資源名と変更後の値をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

rscname_p

値を変更したい資源名(下記の表を参照)。

val_p

変更する資源値が格納された領域へのポインター。例えば、変更する資源が"elapse"の場合、変更後の値(time_t型)を格納した変数を呼び出し側で用意し、それへのポインター(time_t *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

rscname_p	**val_p*	**val_p*の型
"elapse"	ジョブの実行可能時間の制限値 指定できる値の範囲は、1～2147483647秒です。制限しない場合は、PJCMD_UNLIMITEDを指定してください。	time_t
"rscunit"	ジョブを投入するリソースユニット名	char *
"rscgrp"	ジョブを投入するリソースグループ名	char *

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブのパラメーター変更用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*rscname_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

**val_p*が不正です。

- 指定方法が間違っています。
- パラメーター"rscunit"または"rscgrp"の値として指定された値(文字列)がNULLです。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.7.5 pjcmd_alter_get_job_resource()

```
pjcmd_result_t pjcmd_alter_get_job_resource(const PjcmdHandle_t *handle_p, char *rscname_p, void *val_p)
```

ジョブのパラメーターの変更用ハンドルに設定されている、変更するジョブ資源の量を参照します。

[引数]

handle_p

ハンドルへのポインター

rscname_p

参照する資源名。指定できる値は、関数pjcmd_alter_put_job_resource()と同じです。

val_p

*rscuname_p*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブのパラメーター変更用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*rscname_p*または*val_p*が不正(NULL)です。

PJCMD_ERROR_NODATA

指定した資源は、ハンドルに設定されていません。

B.7.6 pjcmd_alter_put_sched_param()

```
pjcmd_result_t pjcmd_alter_put_sched_param(PjcmdHandle_t *handle_p, pjcmd_alter_sched_param_t param, const void *val_p)
```

ジョブのスケジューリングに関して変更するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

変更するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_ALTER_SCHED_PRIORITY	同一ユーザーのジョブにおける変更後の優先度。 0から255までの整数が指定できます。	int
PJCMD_ALTER_SCHED_APRIORITY	リソースユニット内のジョブにおける変更後の優先度。 0から255までの整数が指定できます。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブのパラメーター変更用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法が間違っています。
- 値が間違っています。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

B.7.7 pjcmd_alter_get_sched_param()

```
pjcmd_result_t pjcmd_alter_get_sched_param(const PjcmdHandle_t *handle_p, pjcmd_alter_sched_param_t param, void *val_p)
```

ジョブのパラメーター変更用のハンドルに設定されている、スケジューリングに関するパラメーターの値を参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_alter_put_sched_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブのパラメーター変更用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

B.7.8 pjcmd_alter_set_callback()

```
pjcmd_result_t pjcmd_alter_set_callback(
    PjcmdHandle_t *handle_p,
    void (*alter_wait_callback_func_p)(void),
    void (*alter_accept_callback_func_p)(void))
```

ジョブのパラメーター変更依頼の進行に合わせて、特定のタイミングで呼び出されるコールバック関数を登録します。利用者が特定のタイミングで独自の処理を呼び出したい場合に利用します。例えば、ジョブのパラメーター変更依頼の受付け待ちを示すメッセージを出力するために利用できます。
コールバック関数としてNULLポインターが指定された場合は、そのコールバック関数は設定されていないとみなします。

[引数]

handle_p

ハンドルへのポインター

alter_wait_callback_func_p

ジョブのパラメーター変更依頼の受付け待ちになったときに呼び出す関数へのポインター。この関数を登録すると、ジョブのパラメーター変更依頼の受付けが完了するまで3秒間隔で呼び出されます。

alter_accept_callback_func_p

ジョブのパラメーター変更依頼の受付けが完了したときに呼び出す関数へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブのパラメーター変更用のハンドルではありません。

B.7.9 pjcmd_alter_execute()

```
PjcmdResp_t *pjcmd_alter_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、ジョブのパラメーター変更をジョブ運用管理機能に依頼します。この関数は、ログインノード、システム管理ノード、および計算クラスタ管理ノードで呼び出せます。
変更するパラメーターとして、リソースユニット内のジョブの優先度がハンドルに設定されている場合は、管理者(root)権限が必要です。

[引数]

handle_p

ハンドルへのポインター

[返り値]

ジョブパラメーター変更の応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。ジョブのパラメーター変更の依頼が失敗した場合は、NULLを返し、原因がpjcmd_errcodeに設定されます。
なお、応答情報は依頼が成功したか否かを示します。ジョブのパラメーター変更がジョブ運用管理機能に受け付けられたか否かは、関数pjcmd_get_jobresult_info()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブのパラメーター変更用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノード、システム管理ノード、および計算クラスタ管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。または、リソースユニット内のジョブ優先度の変更を、管理者権限以外で行おうとしました。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_INTERNAL

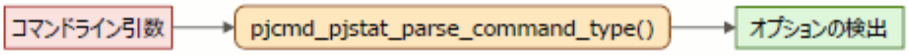
内部エラー

付録C　リファレンス:情報取得API

C.1　情報取得API共通

ここでは、ジョブの情報や資源の情報の取得における、ユーティリティ関数を説明します。

図C.1 pjstatコマンドの引数の検出



C.1.1　pjcmd_pjstat_parse_command_type()

```
pjcmd_pjstat_command_type_t pjcmd_pjstat_parse_command_type(int argc, const char * const *argv_pp)
```

コマンドライン引数をpjstatコマンドの引数として解析し、--rscオプション、--limitオプションおよび--helpオプションの有無を判断します。この関数は、コマンドライン引数に指定されたpjstatコマンドのオプションに応じて、ジョブの情報取得に関するAPI、資源情報取得API、資源状態取得API、およびusageの表示を呼び分けるために使います。

[引数]

argc

引数の個数

argv_pp

引数の配列

[返り値]

PJCMD_PJSTAT_SHOW_JOB

コマンド引数は、ジョブ情報の表示(pjstat)に該当します。

PJCMD_PJSTAT_SHOW_RESOURCE

コマンド引数は、ジョブ用資源情報の表示(pjstat --rsc)に該当します。

PJCMD_PJSTAT_SHOW_LIMIT

コマンド引数は、ジョブ投入時の制限値の表示に(pjstat --limit)に該当します。

PJCMD_PJSTAT_SHOW_HELP

コマンド引数は、usageの表示(pjstat --help)に該当します。

PJCMD_PJSTAT_UNKNOWN_COMMAND_TYPE

コマンド引数からは判断できませんでした。

[pjcmd_errcode]

PJCMD_SUCCESS

成功しました。返り値がPJCMD_PJSTAT_SHOW_JOB、PJCMD_PJSTAT_SHOW_RESOURCE、PJCMD_PJSTAT_SHOW_LIMIT、PJCMD_PJSTAT_SHOW_HELPの場合に設定されます。

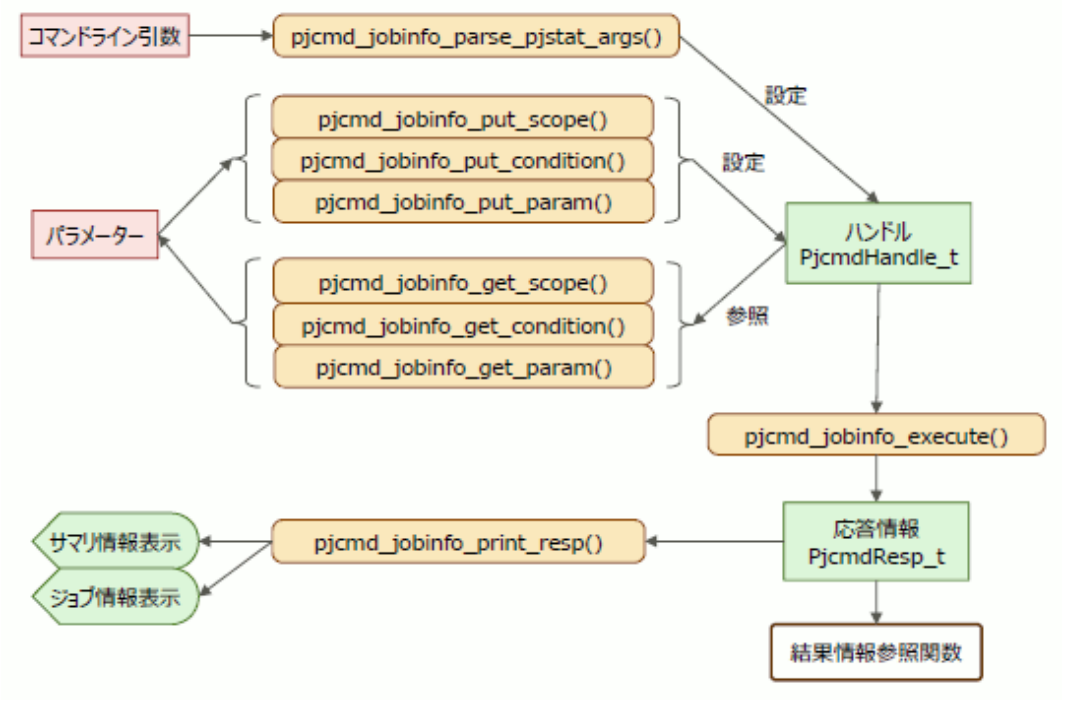
PJCMD_ERROR_UNKNOWN_OPTION

コマンド引数からは判断できませんでした。返り値がPJCMD_PJSTAT_UNKNOWN_COMMAND_TYPEの場合に設定されます。

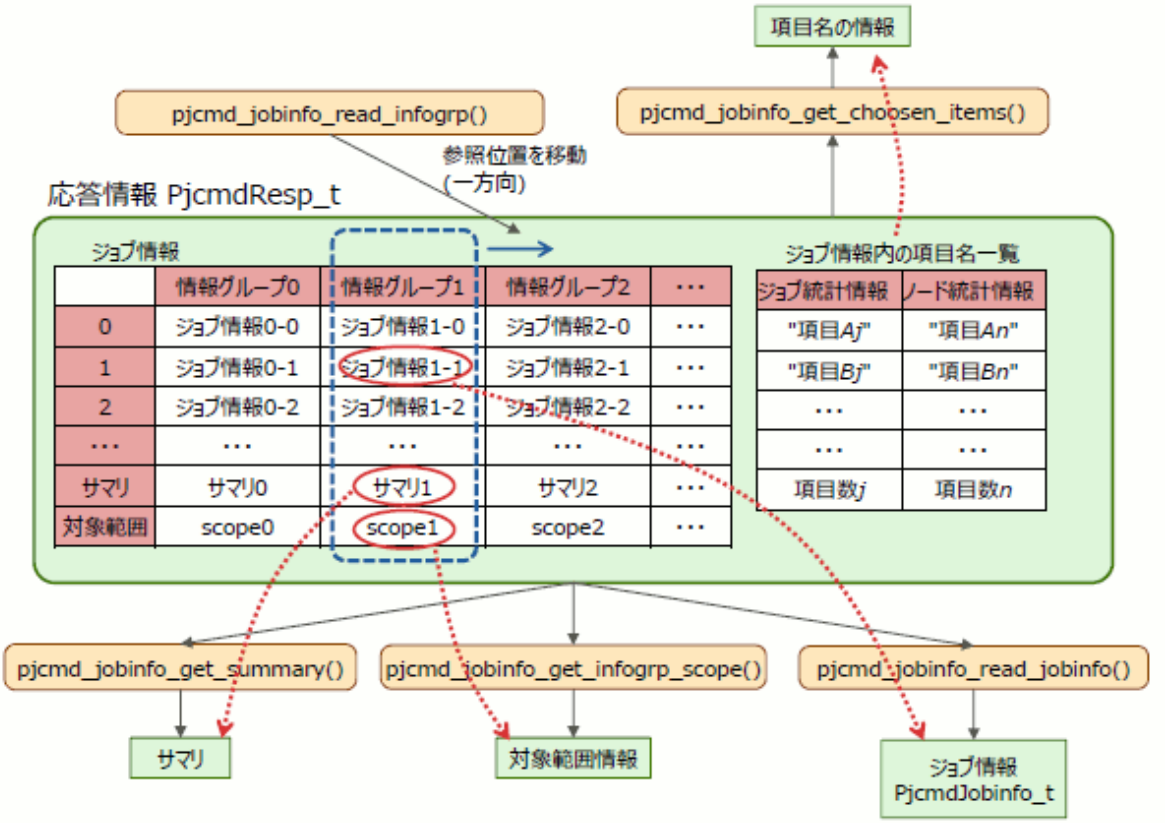
C.2　ジョブの情報取得

ここでは、ジョブの情報(ジョブ統計情報)を取得するための関数を説明します。

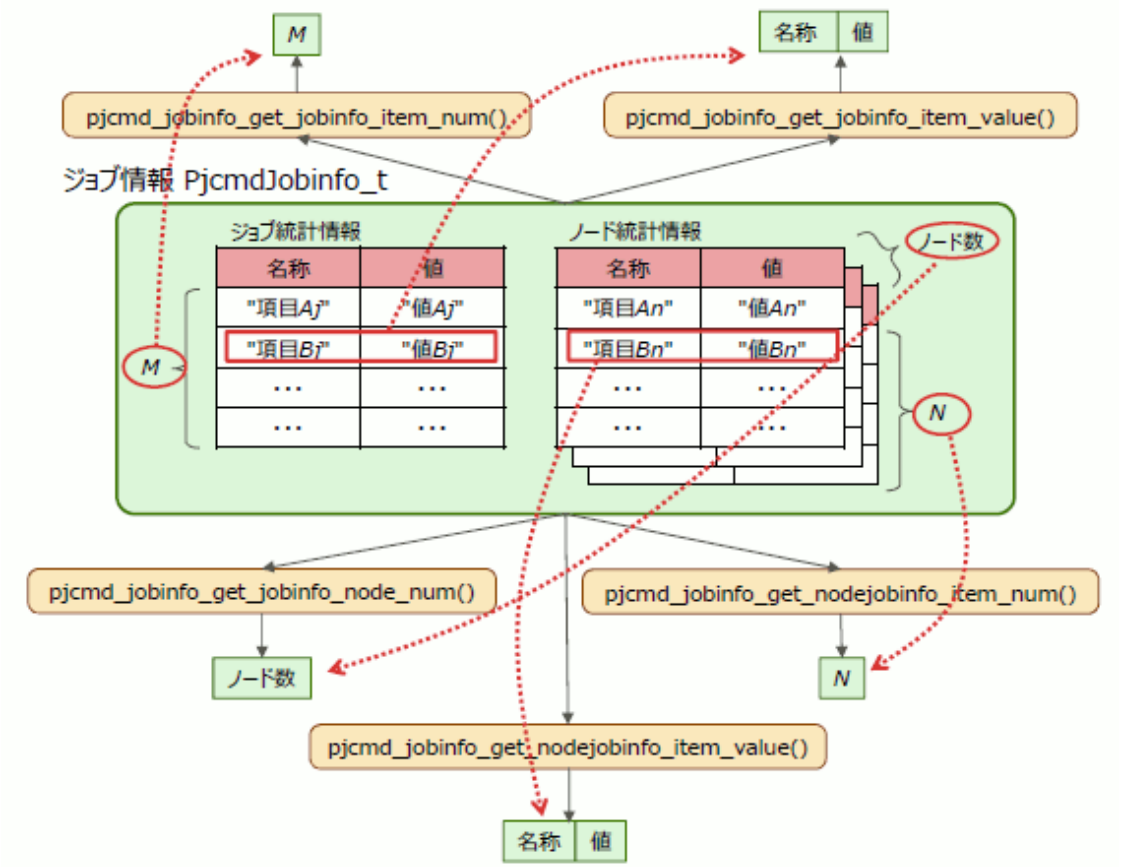
図C.2 ジョブ情報の取得依頼



図C.3 情報グループの参照とジョブ情報の取得



図C.4 ジョブ情報の参照



C.2.1 pjcmd_jobinfo_parse_pjstat_args()

```
pjcmd_result_t pjcmd_jobinfo_parse_pjstat_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjstatコマンドのオプション仕様に従って解析し、指定内容をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。

- ジョブの情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションでない引数が 配列*argv_pp*[]の後ろに移動します。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1]がそのオプションを指します。
なお、本関数は、pjstatコマンドの、--rscオプションや--limitオプション、およびそれらとだけ指定できるオプションについては不正なオプションとみなします。

C.2.2 pjcmd_jobinfo_put_scope()

```
pjcmd_result_t pjcmd_jobinfo_put_scope(PjcmdHandle_t *handle_p, pjcmd_scope_t scope, const void *val_p, uint32_t n)
```

ジョブの情報取得対象範囲をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

scope

情報の取得対象範囲の種類を示す識別子(下記の表を参照)

val_p

ジョブの情報取得対象を示す値が格納された領域へのポインター。例えば、設定する値の型がchar *型の場合、char *型の値を格納した領域を呼び出し側で用意し、それへのポインター(char **)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

n

*val_p*の要素数

scope	**val_p*	**val_p*の型
PJCMD_SCOPE_CLUSTER	クラスタ名(1つだけ。pjstat -c相当)。 設定をしない場合は、環境変数PXMYCLSTの値が適用されます。それも設定されていない場合は、関数pjcmd_jobinfo_execute()がエラーになります。 このパラメーターは、システム管理ノードで呼び出された場合だけ有効になります。システム管理ノード以外では、設定は無視され、呼び出しノードが属するクラスタが適用されます。 また、このパラメーターに対しては、*n*には1を指定してください。	char *
PJCMD_SCOPE_RSCUNIT	リソースユニットごとの情報を取得し、そのリソースユニット名を配列で指定します(要素数*n*。pjstat --rscunit相当)。	char **

scope	**val_p*	**val_p*の型
	リソースユニット名として"*"を指定した場合は、すべてのリソースユニットが対象になります。	
PJCMD_SCOPE_RSCGRP	リソースグループごとの情報を取得し、そのリソースグループ名を配列で指定します(要素数*n*。pjstat --rscgrp相当)。 リソースグループ名として"*"を指定した場合は、すべてのリソースグループが対象になります。	char **

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの情報取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値または指定できない値が指定されました。

PJCMD_ERROR_INVALID_PARAM

*val_p*または*n*が不正です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

C.2.3 pjcmd_jobinfo_get_scope()

```
pjcmd_result_t pjcmd_jobinfo_get_scope(const PjcmdHandle_t *handle_p, pjcmd_scope_t scope, void *val_p, uint32_t *n_p)
```

ジョブの情報取得用のハンドルに設定されている、情報取得対象範囲を参照します。

[引数]

handle_p

ハンドルへのポインター

scope

参照する情報取得対象の識別子。指定できる識別子は、関数pjcmd_jobinfo_put_scope()と同じです。

val_p

*scope*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

n_p

**n_p*に*val_p*の要素数が格納されます。領域は呼び出し側で用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*または*n_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定した*scope*は、ハンドルに設定されていません。

C.2.4 pjcmd_jobinfo_put_condition()

```
pjcmd_result_t pjcmd_jobinfo_put_condition(PjcmdHandle_t *handle_p, pjcmd_jobinfo_condition_type_t type, const char
**item_pp, int n)
```

取得するジョブ情報の条件をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

type

ジョブ情報の取得条件の種類(下記の表を参照)

item_pp

取得条件の項目(文字列)の配列。指定する値は引数*type*に応じて異なります(下記の表を参照)。
NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

n

配列*item_pp*[]の要素数

type	*item_pp*[]
PJCMD_JOBINFO_GROUPING	情報をまとめる単位(取得する単位)を指定します。 "usr" : ユーザー単位でまとめます。 "grp" : グループ単位でまとめます。 "rscu" : リソースユニット単位でまとます。 "rscg" : リソースグループ単位でまとます。 これらは、配列*item_pp*[]として複数を指定できます。 例: "usr"と"rscg"を指定した場合は、同じユーザー、リソースグループごとに情報をまとめるという意味になります。 このパラメーターを指定しなかった場合は、取得する情報は特定の単位ではまとめません。なお、関数pjcmd_jobinfo_put_scope()で、リソースユニットまたはリソースグループを指定している場合は、それぞれ"rscu"、"rscg"が指定されたものとみなします。

type	*item_pp*[]
PJCMD_JOBINFO_CHOOSE PJCMD_JOBINFO_NOT_CHOOSE	ジョブ情報(*)のうち、取得する情報を選択します。 (*) コマンドAPIが取得できる情報は、pjstatコマンドが出力できる情報と同じです(pjstatsinfo(7)参照)。 パラメーターPJCMD_JOBINFO_CHOOSEでは、取得する情報を指定します(pjstat --choose相当)。パラメーターPJCMD_JOBINFO_NOT_CHOOSEでは、取得しない情報を指定します。指定できるのはどちらか一方です。どちらのパラメーターも指定しない場合は、すべての情報を取得します。 対象とする情報は、その項目名を配列*item_pp*[]に指定します。ただし、指定できるのはジョブ情報のうち、"表C.1 ジョブ情報の取得条件で指定できる項目"に示す項目だけです。項目名は複数指定できます。
PJCMD_JOBINFO_FILTER	指定した項目が特定の値のジョブ情報だけ取得します(pjstat --filter相当)。 条件は、配列*item_pp*[]で複数指定でき、1つの配列要素に"*項目*=*値*"の形式で指定します。"*項目*"には、"表C.1 ジョブ情報の取得条件で指定できる項目"に示す項目名が指定できます。 "*値*"には、ワイルドカードや範囲指定表現が使用できます(pjstatコマンドの--filterオプションの書式に準拠)。 なお、関数pjcmd_jobinfo_put_scope()で、リソースユニットやリソースグループを指定している場合は、対応する項目(rscu、rscg)が指定されたものとみなします。
PJCMD_JOBINFO_SORT	取得する情報の並べ替え条件を指定します(pjstat --sort相当)。 条件は、配列*item_pp*[]で複数指定でき、1つの配列要素に"*項目*:*順序*"の形式で並べ替えの条件を指定します。"*項目*"には、"表C.1 ジョブ情報の取得条件で指定できる項目"に示す項目名が指定できます。"*順序*"には、"A"(昇順)または"B"(降順)が指定できます。指定を省略した場合は、ジョブID順に並べられます。

ジョブ情報の取得条件(PJCMD_JOBINFO_CHOOSE、PJCMD_JOBINFO_NOT_CHOOSE、PJCMD_JOBINFO_FILTER、およびPJCMD_JOBINFO_SORT)で指定できる項目名は以下です。

表C.1 ジョブ情報の取得条件で指定できる項目

項目名	意味
jid	ジョブIDまたはサブジョブID
snum	バルクジョブまたはステップジョブのサブジョブ数
jnam	ジョブ名
jtyp	ジョブタイプ
jmdl	ジョブモデル
usr	ジョブの実行ユーザー名
grp	ジョブの実行グループ名
rscu	リソースユニット
rscg	リソースグループ
pri	ジョブの優先度
sh	シェルのパス
mail	メール送信フラグ
adr	メール送信先アドレス
sde	ステップジョブ依存関係式
mask	ジョブ投入ユーザーのumask値
std	標準出力ファイルのパス
stde	標準エラー出力ファイルのパス

項目名	意味
infop	統計情報ファイルのパス
pcl	プロセス単位のCPU使用時間の制限値
pcfl	プロセス単位のコアファイル制限値
pcpl	プロセス単位の最大ユーザープロセス数制限値
pdl	プロセス単位のデータセグメント制限値
prml	プロセス単位のロックメモリサイズ制限値
pmql	プロセス単位のPOSIXメッセージキューサイズ制限値
pofl	プロセス単位のファイルディスクリプタ制限値
ppsl	プロセス単位のシグナル数制限値
ppl	プロセス単位のファイルサイズ制限値
psl	プロセス単位のスタックセグメント制限値
pvml	プロセス単位の仮想メモリサイズ制限値
cmt	ジョブのコメント
rnum	ジョブのリトライ回数
lst	ジョブの以前の状態
st	ジョブの現在の状態
adt	ジョブの投入時刻
qdt	QUEUED状態への遷移時刻
sdt	ジョブ実行の開始時刻
edt	ジョブ実行の終了時刻
exc	EXIT/CANCEL状態への遷移時刻
thldtm	ジョブが固定状態になった累積時間
lhusr	ジョブを固定したユーザー名
holnm	ジョブが固定された回数
prmdt	ジョブ資源管理機能が、各ノードからデータを収集した時刻
ec	シェルスクリプトの終了コード
sn	シグナル番号
pc	PJMコード
ermsg	エラーメッセージ (REASON)
elpl	ジョブの実行経過時間制限値 (実行可能時間)
elp	ジョブの実行経過時間
uctmut	ジョブのユーザーCPU時間の合計
sctmut	ジョブのシステムCPU時間の合計
usctmut	ジョブのユーザーCPU時間とシステムCPU時間の合計
mszl	ノード単位の物理メモリ量制限値
msza	ノード単位の割り当て物理メモリ量
mmszu	各ノードの物理メモリ最大使用量の合計
cnumr	ジョブが要求したCPU数
cnumat	ジョブに対して各ノードで割り当てたCPU数の合計

項目名	意味
cnumut	ジョブが各ノードで使用したCPU数の合計
nnumr	ジョブが要求したノードの数と形状
nnuma	ジョブに割り当てられたノードの数と形状
nnumu	ジョブが使用したノード数
nnumv	ジョブに割り当てられたノードのうち、使用できないノード数
nidlu	ジョブが使用したノードのノードIDリスト
tofulu	ジョブが使用したノードのTofu座標リスト

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの情報取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*type*に不明な値が指定されました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

C.2.5 pjcmd_jobinfo_get_condition()

```
pjcmd_result_t pjcmd_jobinfo_get_condition(const PjcmdHandle_t *handle_p, pjcmd_jobinfo_condition_type_t type, char ***item_ppp, int *n_p)
```

ジョブの情報取得用のハンドルに設定されている、情報の取得条件の内容を参照します。

[引数]

handle_p

ハンドルへのポインター

type

参照するジョブ情報の取得条件を示す識別子。指定できる識別子は、関数pjcmd_jobinfo_put_condition()と同じです。

item_ppp

*type*で指定した情報取得条件の項目名が、文字列配列(**item_ppp*)[]に格納されます。(**item_ppp*)[]は、ハンドル内に設定されている領域を指します。

n_p

n_p*に文字列配列(item_ppp*)[]の要素数が格納されます。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*item_ppp*または*n_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*type*に不明な値が指定されました。

PJCMD_ERROR_NODATA

*type*で指定された取得条件は設定されていません。

C.2.6 pjcmd_jobinfo_put_param()

```
pjcmd_result_t pjcmd_jobinfo_put_param(PjcmdHandle_t *handle_p, pjcmd_jobinfo_param_t param, void *val_p)
```

ジョブの情報取得に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定するジョブ情報取得に関するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_JOBINFO_USER_ID	情報を取得する対象ユーザーIDの配列を指定します。配列の最後の要素は-1にしてください。	uid_t *
PJCMD_JOBINFO_USER_NAME	情報を取得する対象ユーザー名を指定します。配列の最後の要素はNULLにしてください。	char **
PJCMD_JOBINFO_GROUP_ID	情報を取得する対象グループIDを指定します。配列の最後の要素は-1にしてください。	gid_t *
PJCMD_JOBINFO_GROUP_NAME	情報を取得する対象グループ名を指定します。配列の最後の要素はNULLにしてください。	char **
PJCMD_JOBINFO_HISTORY_DAY	過去何日間に終了したジョブの情報を取得するかを指定します(pjstat -Hおよび-H day=*n*相当) 値には1から365までの整数が指定できます。-1を指定すると、過去3日間に終了したジョブの情報を取得します(pjstatコマンドの-Hオプションの引数がない場合に相当)。	int
PJCMD_JOBINFO_HISTORY_START	取得する終了ジョブの期間の開始日を指定します(pjstat -H start=*date*相当)。	time

param	**val_p*	**val_p*の型
PJCMD_JOBINFO_HISTORY_END	取得する終了ジョブの期間の終了日を指定します(pjstat -H end=*date*相当)。	time
PJCMD_JOBINFO_HISTORY_PERIOD	パラメーターPJCMD_JOBINFO_HISTORY_STARTで指定した開始日を1日目として、以降何日間の終了ジョブを取得対象とするかを指定します(pjstat -H period=*days*相当) 指定できる値の範囲は、1～2147483647日間です。	int
PJCMD_JOBINFO_VERBOSITY	取得する情報の粒度を指定します。 ・ PJCMD_JOBINFO_VERBOSITY_JOB ジョブの情報だけを取得します(デフォルト)。 ・ PJCMD_JOBINFO_VERBOSITY_SUBJOB ジョブおよびサブジョブの情報を取得します(pjstat -E相当)。	int
PJCMD_JOBINFO_SUMMARY	ジョブ情報取得時にサマリ情報を取得するか否かを指定します。 ・ PJCMD_JOBINFO_NO_SUMMARY サマリ情報は取得しません(デフォルト)。 ・ PJCMD_JOBINFO_WITH_SUMMARY サマリ情報、ジョブ情報の両方を取得します(pjstat --with-summary相当) ・ PJCMD_JOBINFO_ONLY_SUMMARY サマリ情報だけ取得します(pjstat --summary相当)	int
PJCMD_JOBINFO_LEVEL	取得する情報のレベル。 ・ PJCMD_JOBINFO_LEVEL_0 基本情報だけ取得します(pjstat 相当、デフォルト)。 ・ PJCMD_JOBINFO_LEVEL_1 基本情報と追加情報を取得します(pjstat -v相当)。 ・ PJCMD_JOBINFO_LEVEL_2 ジョブ統計情報を取得します(pjstat -s相当)。 ・ PJCMD_JOBINFO_LEVEL_3 ジョブ統計情報とノード統計情報を取得します(pjstat -S相当)。	int
PJCMD_JOBINFO_LEVEL_PATTERN	取得する情報のレベルがPJCMD_JOBINFO_LEVEL_1の場合に、さらに追加して取得する情報を指定します(pjstat -v pattern=*n*相当) 現在、指定できる値は1だけです。	uint32_t
PJCMD_JOBINFO_OTHERSJOB	ほかのユーザー(本関数を呼び出しているユーザー以外)が投入したジョブ情報を取得するか否かを指定します(pjstat -A相当)。 ・ PJCMD_JOBINFO_OTHERSJOB_NONE ほかのユーザーのジョブ情報は取得しません(デフォルト)。 ・ PJCMD_JOBINFO_OTHERSJOB_ALL ほかのユーザーのジョブ情報も取得します。 ただし、権限がない場合は、一部の情報は参照できません。	int

param	**val_p*	**val_p*の型
PJCMD_JOBINFO_DATA	pjstatコマンドの--dataオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、取得するジョブ情報には影響しません。	int
PJCMD_JOBINFO_DELIMITER	pjstatコマンドの--delimiterオプションの指定に相当。 パラメーターPJCMD_JOBINFO_DATAが指定されている場合に、関数pjcmd_jobinfo_print_resp()でジョブ情報を表示する際の区切り文字を指定します。このパラメーターを設定しない場合は、区切り文字はコンマ(,)が使用されます。 このパラメーターは取得するジョブ情報には影響しません。	char *
PJCMD_JOBINFO_HELP	pjstatコマンドの--helpオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、取得するジョブ情報には影響しません。	int

パラメーターPJCMD_JOBINFO_USER_IDとPJCMD_JOBINFO_USER_NAME、およびPJCMD_JOBINFO_GROUP_IDとPJCMD_JOBINFO_GROUP_NAMEは、それぞれ最後に指定されたものが有効になります。また、それらを指定しない場合は、関数を呼び出したユーザー、グループが適用されます。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの情報取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値または指定できない値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法の間違い
- 間違った値

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

C.2.7 pjcmd_jobinfo_get_param()

```
pjcmd_result_t pjcmd_jobinfo_get_param(const PjcmdHandle_t *handle_p, pjcmd_jobinfo_param_t param, void *val_p)
```

ジョブの情報取得用のハンドルに設定されている、情報取得に関するパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_jobinfo_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブの情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

C.2.8 pjcmd_jobinfo_execute()

```
PjcmdResp_t *pjcmd_jobinfo_execute(PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、ジョブの情報取得をジョブ運用管理機能に依頼します。

[引数]

handle_p

ハンドルへのポインター

[返り値]

ジョブ情報取得の応答情報。
得られた応答情報は、pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。ジョブの情報取得の依頼が失敗した場合は、NULLを返し、原因がpjcmd_errcodeに設定されます。
なお、応答情報は依頼が成功したか否かを示します。ジョブの情報が正常に取得できたかどうかは、関数pjcmd_get_result()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。

- ジョブの情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノード、計算クラスタ管理ノード、およびシステム管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_INTERNAL

内部エラー

C.2.9 pjcmd_jobinfo_get_choosen_item()

```
pjcmd_result_t pjcmd_jobinfo_get_choosen_item(const PjcmdResp_t *resp_p, char ***jobinfo_item_ppp, int *jobinfo_item_n_p,
char ***nodeinfo_item_ppp, int *nodeinfo_item_n_p)
```

ジョブ情報取得結果に含まれている情報の項目名の一覧を取得します。

[引数]

resp_p

応答情報へのポインター

jobinfo_item_ppp

ジョブ情報の項目名一覧が文字列配列(**jobinfo_item_ppp*)[]に格納されます。格納される項目名は"表C.2 ジョブ情報の項目名、名称、および値(1)"を参照してください。

jobinfo_item_n_p

jobinfo_item_n_p*に、ジョブ情報の項目名一覧を格納した配列(jobinfo_item_ppp*)[]の要素数が格納されます。

nodeinfo_item_ppp

ノードごとのジョブ情報の項目名一覧が文字列配列(**nodeinfo_item_ppp*)[]に格納されます。格納される項目名は"表C.3 ジョブ情報の項目名、名称、および値(2)"を参照してください。

nodeinfo_item_n_p

*node*info_item_n_p*に、ノードごとのジョブ情報の項目名一覧を格納した配列(**nodeinfo_item_ppp*)[]の要素数が格納されます。

*jobinfo_item_ppp*と*nodeinfo_item_ppp*が指す領域は、応答情報内に確保された領域であり、呼び出し側が直接解放しないでください。この領域は、応答情報を解放するまでは保持されます。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- ジョブの情報取得の応答情報ではありません。

PJCMD_ERROR_INVALID_ARGUMENT

jobinfo_item_ppp、*jobinfo_item_n_p*、*nodeinfo_item_ppp*、または*nodeinfo_item_n_p*が不正(NULL)です。

C.2.10 pjcmd_jobinfo_read_infogrp()

```
pjcmd_result_t pjcmd_jobinfo_read_infogrp(PjcmdResp_t *resp_p)
```

応答情報に格納されている情報の取得単位である情報グループについて、参照位置を次の情報グループに移動します。

[引数]

resp_p

応答情報へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- ジョブの情報取得の応答情報ではありません。

PJCMD_ERROR_NODATA

次の情報グループはありません(すべて参照しました)。

PJCMD_ERROR_INVALID_PARAM

応答情報内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

C.2.11 pjcmd_jobinfo_print_resp()

```
pjcmd_result_t pjcmd_jobinfo_print_resp(PjcmdResp_t *resp_p, pjcmd_jobinfo_print_type_t type)
```

参照の対象としている情報グループに含まれるジョブの情報を、pjstatコマンドの仕様に従って標準出力に出力します。

対象の情報グループに対して以下の操作をしている場合はエラーになります。

- この関数を2回以上呼び出した場合
- この関数を呼び出す前に関数pjcmd_jobinfo_read_jobinfo()で情報グループを参照している場合

この関数の呼び出し後は対象の情報グループ内のジョブ情報は最後まで参照された状態になります。したがって、この関数を呼び出した後に同じ情報グループに対して関数pjcmd_jobinfo_read_jobinfo()を呼び出しても情報は取得できません。

[引数]

resp_p

応答情報へのポインター

type

出力内容の種類。

PJCMD_JOBINFO_PRINT_SUMMARY

ジョブのサマリ情報を表示します。

PJCMD_JOBINFO_PRINT_JOBINFO

ジョブの詳細情報を表示します。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- ジョブの情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*type*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

関数pjcmd_jobinfo_put_scope()やpjcmd_jobinfo_put_param()で指定した情報の取得範囲や条件の指定が正しくないため、情報が取得できませんでした。
または、この関数を呼び出す前に、関数pjcmd_jobinfo_read_jobinfo()によって情報グループ内のジョブ情報が参照されています。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

C.2.12 pjcmd_jobinfo_get_summary()

```
pjcmd_result_t pjcmd_jobinfo_get_summary(const PjcmdResp_t *resp_p, pjcmd_job_status_t status, int64_t *num_p)
```

情報グループに含まれるジョブのうち、特定の状態にあるジョブ数とサブジョブ数を参照します。
関数pjcmd_jobinfo_put_param()で、情報の粒度(PJCMD_JOBINFO_VERBOSITY)としてPJCMD_JOBINFO_VERBOSITY_SUBJOBを指定していない場合、サブジョブ数は0になります。また、関数pjcmd_jobinfo_put_param()で、PJCMD_JOBINFO_HISTORYを指定していない場合は、終了ジョブの数は取得できず、エラーになります。

[引数]

resp_p

応答情報へのポインター

status

参照する対象ジョブの状態の識別子(下記の表を参照)。

status	説明
PJCMD_JOB_STATUS_ACCEPT	投入が受け付けられた状態のジョブ(ACCEPT状態)
PJCMD_JOB_STATUS_QUEUED	実行待ちのジョブ(QUEUED状態)
PJCMD_JOB_STATUS_RUNNING_A	実行依頼中のジョブ(RUNNING-A状態)
PJCMD_JOB_STATUS_RUNNING_P	プロローグ処理中のジョブ(RUNNING-P状態)
PJCMD_JOB_STATUS_RUNNING	実行中のジョブ(RUNNING状態)
PJCMD_JOB_STATUS_RUNNING_E	エピローグ処理中のジョブ(RUNNING-E状態)
PJCMD_JOB_STATUS_RUNOUT	終了処理完了待ちのジョブ(RUNOUT状態)
PJCMD_JOB_STATUS_HOLD	ユーザーによって固定されたジョブ(HOLD状態)
PJCMD_JOB_STATUS_ERROR	エラーによって固定されたジョブ(ERROR状態)
PJCMD_JOB_STATUS_SUSPEND	サスペンド処理中のジョブ(SUSPEND状態)
PJCMD_JOB_STATUS_SUSPENDED	サスペンド済みのジョブ(SUSPENDED状態)
PJCMD_JOB_STATUS_RESUME	リジューム処理中のジョブ(RESUME状態)
PJCMD_JOB_STATUS_REJECT	投入が受け付けれらなかったジョブ(REJECT状態)
PJCMD_JOB_STATUS_EXIT	終了したジョブ(EXIT状態)
PJCMD_JOB_STATUS_CANCEL	実行が中止されたジョブ(CANCEL状態)
PJCMD_JOB_STATUS_ALL	終了していないすべてのジョブ (ACCEPT、QUEUED、RUNNING_A、RUNNING_P、RUNNING、RUNNING_E、RUNOUT、HOLD、ERROR、SUSPEND、SUSPENDED、およびRESUME状態)
PJCMD_JOB_STATUS_END	終了したジョブ(REJECT、EXIT、およびCANCEL状態)
PJCMD_JOB_STATUS_TOTAL	すべてのジョブ (ACCEPT、QUEUED、RUNNING_A、RUNNING_P、RUNNING、RUNNING_E、RUNOUT、HOLD、ERROR、SUSPEND、SUSPENDED、RESUME、REJECT、EXIT、およびCANCEL状態)

num_p

配列*num_p*[]にジョブ数とサブジョブ数が格納されます。int64_t型の値を2つ格納できる配列領域を呼び出し側で用意する必要があります。

[返り値]

PJCMD_OK

成功。引数*num_p*[0]にジョブ数、*num_p*[1]にサブジョブ数が格納されます。

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- ジョブの情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*num_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*status*に不明な値が指定されました。

PJCMD_ERROR_NODATA

終了ジョブの情報が応答情報に含まれていない場合に、それらの数を取得しようしました。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_INVALID_PARAM

関数pjcmd_jobinfo_put_scope()やpjcmd_jobinfo_put_param()で指定した情報の取得範囲や条件の指定が正しくないため、情報が取得できませんでした。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

C.2.13 pjcmd_jobinfo_get_infogrp_scope()

```
pjcmd_result_t pjcmd_jobinfo_get_infogrp_scope(const PjcmdResp_t *resp_p, char **rscunit_pp, char **rscgrp_pp, char **uname_pp, char **gname_pp)
```

現在参照している情報グループについて、そのリソースユニット名、リソースグループ名、ユーザー名、およびグループ名を取得します。

[引数]

resp_p

応答情報へのポインター

rscunit_pp

**rscunit_pp*にリソースユニット名へのポインターが格納されます。

rscgrp_pp

**rscgrp_pp*にリソースグループ名へのポインターが格納されます。

uname_pp

**uname_pp*にユーザー名へのポインターが格納されます。

gname_pp

**gname_pp*にグループ名へのポインターが格納されます。

rscunit_pp、*rscgrp_pp*、*uname_pp*、および*gname_pp*が指す領域は、応答情報内に確保された領域であり、呼び出し側が直接解放しないでください。この領域は応答情報の解放または次の情報グループを読み込むまで保持されます。

[返り値]

PJCMD_OK

成功。得られる値は、関数pjcmd_jobinfo_put_scope()やpjcmd_jobinfo_put_condition()で指定した取得対象単位(リソースユニット、リソースグループ、ユーザー、グループ)の情報です。このため、指定していない対象についてはその名称は得られず、(char *)NULLが格納されます。

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- ジョブの情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_ARGUMENT

rscunit_pp、*rscgrp_pp*、*uname_pp*、または*gname_pp*が不正(NULL)です。

C.2.14 pjcmd_jobinfo_read_jobinfo()

```
PjcmdJobinfo_t *pjcmd_jobinfo_read_jobinfo(PjcmdResp_t *resp_p)
```

現在参照中の情報グループに含まれる、次のジョブの情報を取得します。
ただし、関数pjcmd_jobinfo_put_param()で指定した情報の粒度(PJCMD_JOBINFO_VERBOSITY)がサマリ情報だけの場合(PJCMD_JOBINFO_VERBOSITY_SUMMARY)は取得できません。

[引数]

resp_p

応答情報へのポインター

[返り値]

ジョブ情報。
失敗した場合はNULLを返し、原因がpjcmd_errcodeに設定されます。返されるジョブ情報は、次のジョブ情報を読み込むと内容は不定になります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。

- ジョブの情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_NODATA

現情報グループには次のジョブ情報はありません。または、取得した情報はサマリ情報だけでなので、ジョブ情報はありません。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_INVALID_PARAM

関数pjcmd_jobinfo_put_scope()やpjcmd_jobinfo_put_param()で指定した情報の取得範囲や条件の指定が正しくないため、情報が取得できませんでした。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

C.2.15 pjcmd_jobinfo_get_jobinfo_item_num()

```
int pjcmd_jobinfo_get_jobinfo_item_num(const PjcmdJobinfo_t *jobinfo_p)
```

ジョブ情報に含まれる情報の項目数を返します。

[引数]

jobinfo_p

ジョブ情報へのポインター

[返り値]

ジョブ情報に含まれる情報の項目数。
値はジョブ情報として取得できるすべての項目数であり、取得していない項目も含みます。
失敗した場合は-1を返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*jobinfo_p*が不正(NULL)です。

C.2.16 pjcmd_jobinfo_get_jobinfo_item_value()

```
pjcmd_result_t pjcmd_jobinfo_get_jobinfo_item_value(PjcmdJobinfo_t *jobinfo_p, int indx, char **name_pp, char **val_pp)
```

ジョブ情報に含まれる各情報の名称とその値を取得します。

[引数]

jobinfo_p

ジョブ情報へのポインター

indx

取得したい情報のインデックス。値は、0から"関数pjcmd_jobinfo_get_jobinfo_item_num()で取得する値-1"の範囲が指定できます。

name_pp

**name_pp*に*Indx*で指定された情報の名称へのポインターが格納されます。格納される名称は"表C.2 ジョブ情報の項目名、名称、および値(1)"を参照してください。

val_pp

**val_pp*に*Indx*で指定された情報の値(文字列)へのポインターが格納されます。格納される値は"表C.2 ジョブ情報の項目名、名称、および値(1)"を参照してください。

*name_pp*および*val_pp*が指す領域は、応答情報内に確保された領域であり、呼び出し側が直接解放しないでください。*name_pp*が指す領域は応答情報の解放まで保持されます。*val_pp*が指す領域は応答情報の解放または本関数を次に呼び出すまで保持されます。

[返り値]

PJCMD_OK

成功。指定した項目についての情報が取得されていない場合は、(char *)**val_pp*にNULLが格納されます。
ジョブ情報取得時のパラメーターPJCMD_JOBINFO_OTHERSJOB(ほかのユーザーのジョブ情報取得)を、PJCMD_JOBINFO_OTHERSJOB_ALL(すべて取得)としている場合、すべてのユーザーのジョブ情報を取得できます。しかし、参照の権限がない情報については(char *)**val_pp*が指す文字列は"?"になります。

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

jobinfo_p、*name_pp*、または*val_pp*が不正(NULL)です。

PJCMD_ERROR_NODATA

*indx*の値が範囲外です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

C.2.17 pjcmd_jobinfo_get_jobinfo_node_num()

```
pjcmd_result_t pjcmd_jobinfo_get_jobinfo_node_num(const PjcmdJobinfo_t *jobinfo_p, uint32_t *num_p)
```

ジョブ情報に含まれる、ノード単位の情報の数(ノード数)を取得します。

[引数]

jobinfo_p

ジョブ情報へのポインター

num_p

**num_p*にノード単位の情報の数(ノード数)が格納されます。
対象となるジョブが、ステップジョブ、バルクジョブのサマリジョブ、または実行前のジョブの場合は、ノード情報がないため、**num_p*には0が格納されます。

[返り値]

PJCMD_OK

成功。

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

ジョブ情報が不正です。*jobinfo_p*がNULL、またはジョブ情報取得のレベルがPJCMD_JOBINFO_LEVEL_3ではありません。

C.2.18 pjcmd_jobinfo_get_nodejobinfo_item_num()

```
int pjcmd_jobinfo_get_nodejobinfo_item_num(const PjcmdJobinfo_t *jobinfo_p)
```

ノード単位のジョブ情報に含まれる情報の項目数を返します。

[引数]

jobinfo_p

ジョブ情報へのポインター

[返り値]

ノード単位のジョブ情報における情報の項目数。
値はジョブ情報として取得できる(保存設定されている)すべての項目数であり、取得していない項目も含みます。ただし、ノード単位のジョブ情報を取得していない場合は、エラーになります。
失敗した場合は-1を返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*jobinfo_p*が不正(NULL)です。

PJCMD_ERROR_NODATA

ジョブ情報には、ノード単位のジョブ情報がありません。

C.2.19 pjcmd_jobinfo_get_nodejobinfo_item_value()

```
pjcmd_result_t pjcmd_jobinfo_get_nodejobinfo_item_value(PjcmdJobinfo_t *jobinfo_p, uint32_t node_indx, int item_indx,
char **name_pp, char **val_pp)
```

ノード単位のジョブ情報に含まれる各情報の名称とその値を取得します。

[引数]

jobinfo_p

ジョブ情報へのポインター

node_indx

ジョブ情報にあるノードのインデックス。値は、0から"関数pjcmd_jobinfo_get_jobinfo_node_num()で得られる値-1"の範囲が指定できます。

item_indx

取得したい情報のインデックス。値は、0から"関数pjcmd_jobinfo_get_nodejobinfo_item_num()で得られる値-1"の範囲が指定できます。

name_pp

**name_pp*に*item_indx*で指定された情報の名称へのポインターが格納されます。格納される名称は"表C.3 ジョブ情報の項目名、名称、および値(2)"を参照してください。

val_pp

**val_pp*に、*item_indx*で指定された情報の値(文字列)のポインターが格納されます。格納される値は"表C.3 ジョブ情報の項目名、名称、および値(2)"を参照してください。

*name_pp*および*val_pp*が指す領域は、応答情報内に確保された領域であり、呼び出し側が直接解放しないでください。*name_pp*が指す領域は応答情報の解放まで保持されます。*val_pp*が指す領域は応答情報の解放または本関数を次に呼び出すまで保持されます。

[返り値]

PJCMD_OK

成功。情報が取得されていない場合は、**val_pp*にNULLが格納されます。

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

nodejobinfo_p、*name_pp*、または*val_pp*が不正(NULL)です。

PJCMD_ERROR_NODATA

node_indx、*item_indx*が範囲外の値です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

C.2.20 ジョブ情報の項目名、名称、および値

以下の関数で得られる情報を"表C.2 ジョブ情報の項目名、名称、および値(1)"に示します。

- 関数pjcmd_jobinfo_get_choosen_item()の引数*jobinfo_item_ppp*に格納される情報の項目名
- 関数pjcmd_jobinfo_get_jobinfo_item_value()の引数*name_pp*に格納される情報の名称
- 関数pjcmd_jobinfo_get_jobinfo_item_value()の引数*val_pp*に格納される情報の値

表C.2 ジョブ情報の項目名、名称、および値(1)

項目名 (*jobinfo_item_ppp*)	名称 (*name_pp*)	値 (*val_pp*)
jid	JOB ID	ジョブIDまたはサブジョブID pjstatコマンドが表示するジョブ統計情報 "JOB ID"と同じ形式です(pjstatsinfo(7)参照)。
snum	SUB JOB NUM	バルクジョブまたはステップジョブのサブジョブ数 pjstatコマンドが表示するジョブ統計情報"SUB JOB NUM"と同じ形式です(pjstatsinfo(7)参照)。
nnumr	NODE NUM (REQUIRE)	ジョブが要求したノードの数と形状 pjstatコマンドが表示するジョブ統計情報"NODE NUM (REQUIRE)"と同じ形式です(pjstatsinfo(7)参照)。
nnuma	NODE NUM (ALLOC)	ジョブに割り当てられたノードの数と形状 pjstatコマンドが表示するジョブ統計情報"NODE NUM (ALLOC)"と同じ形式です(pjstatsinfo(7)参照)。
上記以外	pjstatsinfo(7)に載っているジョブ統計情報のうち、pjstatコマンドで出力される項目と同じです。 例: 項目名が"jnam"の場合、名称は"JOB NAME"、値はジョブ名です。	

以下の関数で得られる情報を"表C.3 ジョブ情報の項目名、名称、および値(2)"に示します。

- 関数pjcmd_jobinfo_get_choosen_item()の引数*nodeinfo_item_ppp*に格納される情報の項目名
- 関数pjcmd_jobinfo_get_nodejobinfo_item_value()の引数*name_pp*に格納される情報の名称
- 関数pjcmd_jobinfo_get_nodejobinfo_item_value()の引数*val_pp*に格納される情報の値

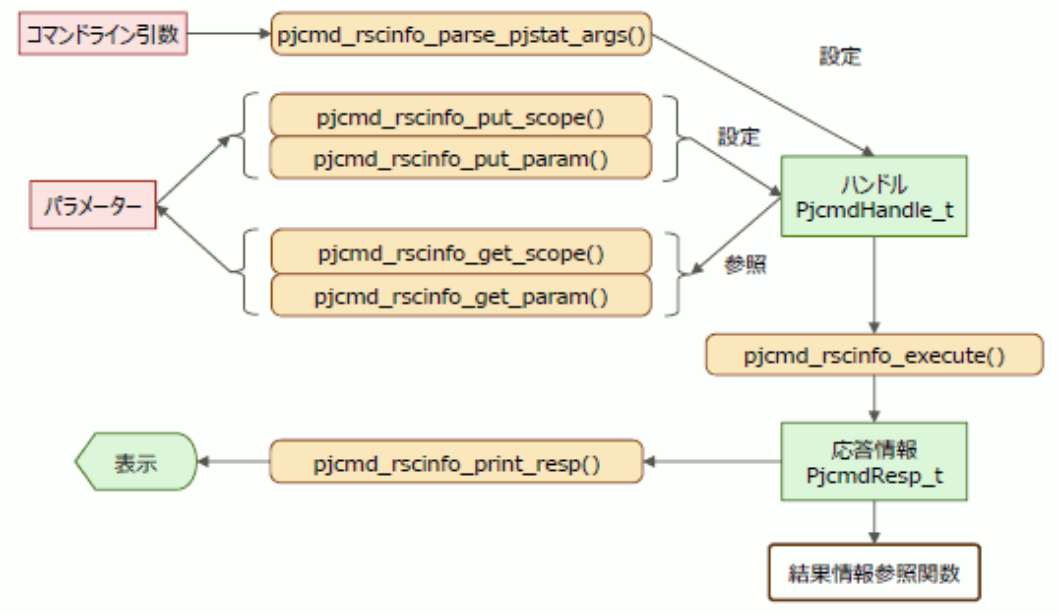
表C.3 ジョブ情報の項目名、名称、および値(2)

項目名 (*nodeinfo_item_ppp*)	名称 (*name_pp*)	値 (*val_pp*)
ntofu	TOFU COORDINATE	ノードのTofu座標 pjstatコマンドが表示するノード/仮想ノード統計情報"TOFU COORDINATE"と同じ形式です(pjstatsinfo(7)参照)。
node	NODE COORDINATE	使用ノードの座標 pjstatコマンドが表示するノード/仮想ノード統計情報"NODE COORDINATE"と同じ形式です(pjstatsinfo(7)参照)。
上記以外	pjstatsinfo(7)に載っているノード/仮想ノード統計情報のうち、pjstatコマンドで出力される項目と同じです。 例: 項目名が"vnid"の場合、名称は"VNODE ID"、値は仮想ノードIDです。	

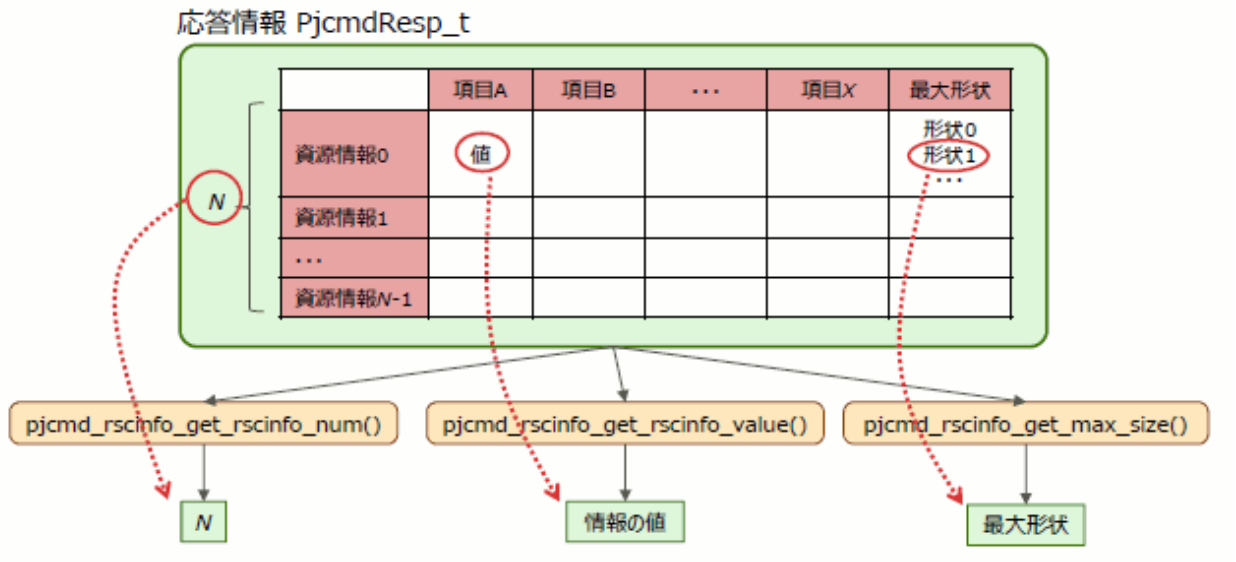
C.3 ジョブ用資源の情報取得

ここでは、ジョブを実行するシステムの資源について情報を取得するための関数を説明します。

図C.5 資源情報の取得依頼



図C.6 資源情報の参照



C.3.1 pjcmd_rscinfo_parse_pjstat_args()

```
pjcmd_result_t pjcmd_rscinfo_parse_pjstat_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjstatコマンドの--rscオプション指定時の仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 資源情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションでない引数が 配列*argv_pp*[]の後ろに移動します。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1] がそのオプションを指します。
なお、本関数は、pjstatコマンドの--rscオプションおよびそれと共に指定できるオプションだけを認識します。

C.3.2 pjcmd_rscinfo_put_scope()

```
pjcmd_result_t pjcmd_rscinfo_put_scope(PjcmdHandle_t *handle_p, pjcmd_scope_t scope, void *val_p, uint32_t n)
```

資源情報の取得対象範囲をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

scope

情報の取得対象範囲の種類を示す識別子(下記の表を参照)

val_p

情報取得対象を示す値が格納された領域へのポインター。例えば、設定する値の型がchar *型の場合、char *型の値を格納した領域を呼び出し側で用意し、それへのポインター(char **)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

n

val_p の要素数

scope	**val_p*	**val_p*の型
PJCMD_SCOPE_CLUSTER	クラスタ名(1つだけ。pjstat -c相当) 設定をしない場合は、環境変数PXMYCLSTの値が適用されます。それも設定されていない場合は、関数pjcmd_rscinfo_execute()がエラーになります。 このパラメーターは、システム管理ノードで呼び出された場合だけ有効になります。システム管理ノード以外では、設定は無視され、呼び出しノードが属するクラスタが適用されます。 また、このパラメーターに対しては、*n*には1を指定してください。	char *
PJCMD_SCOPE_RSCUNIT	リソースユニット名の配列(要素数*n*。(pjstatコマンドの--rscunitに相当) このパラメーターを設定しない場合は、クラスタ内のすべてのリソースユニットが対象になります。	char **
PJCMD_SCOPE_RSCGRP	リソースグループ名の配列(要素数*n*。pjstatコマンドの--rscgrpに相当) このパラメーターを設定しない場合は、クラスタ内のすべてのリソースユニットが対象になります。	char **

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 資源情報取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値または指定できない値が指定されました。

PJCMD_ERROR_INVALID_PARAM

*val_p*または*n*が不正です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

C.3.3 pjcmd_rscinfo_get_scope()

```
pjcmd_result_t pjcmd_rscinfo_get_scope(const PjcmdHandle_t *handle_p, pjcmd_scope_t scope, void *val_p, uint32_t *n_p)
```

ハンドルに設定されている資源情報の取得対象範囲を参照します。

[引数]

handle_p

ハンドルへのポインター

scope

参照する情報取得対象範囲の識別子。指定できる識別子は、関数pjcmd_rscinfo_put_scope()と同じです。

val_p

*scope*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

n_p

**n_p*に*val_p*の要素数が格納されます。領域は呼び出し側で用意する必要があります。

[返り値]

PJCMD_OK

成功。

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 資源情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*または*n*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定した*scope*は、ハンドルに設定されていません。

C.3.4 pjcmd_rscinfo_put_param()

```
pjcmd_result_t pjcmd_rscinfo_put_param(PjcmdHandle_t *handle_p, pjcmd_rscinfo_param_t param, const void *val_p)
```

資源情報取得に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

設定する資源情報取得に関するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_RSCINFO_SHAPE	pjstatコマンドの--shapeオプションの指定に相当。 0: リソースグループの最大形状情報を表示しません(デフォルト)。 1: リソースグループの最大形状情報を表示します。 このパラメーターは、情報の取得内容に影響はせず、関数pjcmd_rscinfo_print_resp()で表示される内容に影響します。	int
PJCMD_RSCINFO_HELP	pjstatコマンドの--helpオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、取得するジョブ情報には影響しません。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 資源情報取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法の間違い
- 間違った値

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

C.3.5 pjcmd_rscinfo_get_param()

```
pjcmd_result_t pjcmd_rscinfo_get_param(const PjcmdHandle_t *handle_p, pjcmd_rscinfo_param_t param, void *val_p)
```

ハンドルに設定されている、資源情報取得に関するパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、pjcmd_rscinfo_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 資源情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

C.3.6 pjcmd_rscinfo_execute()

```
PjcmdResp_t* pjcmd_rscinfo_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、資源情報の取得をジョブ運用管理機能に依頼します。

[引数]

handle_p

ハンドルへのポインター

[返り値]

リソースユニット・リソースグループの資源情報取得の応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。情報の取得の依頼が失敗した場合は、NULLを返し、pjcmd_errcodeが設定されます。
なお、応答情報は依頼が成功したか否かを示します。情報が正常に取得できたかどうかは、関数pjcmd_get_result()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 資源情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノード、計算クラスタ管理ノード、およびシステム管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_INTERNAL

内部エラー

C.3.7 pjcmd_rscinfo_print_resp()

```
pjcmd_result_t pjcmd_rscinfo_print_resp(const PjcmdResp_t *resp_p)
```

資源情報の取得結果を、pjstatコマンドの--rscオプション指定時の仕様に従って標準出力に出力します。

[引数]

resp_p

応答情報へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

C.3.8 pjcmd_rscinfo_get_rscinfo_num()

```
int pjcmd_rscinfo_get_rscinfo_num(const PjcmdResp_t *resp_p)
```

資源情報取得の応答情報に含まれる資源情報の数を取得します。

[引数]

resp_p

応答情報へのポインター

[返り値]

資源情報数。
失敗した場合は-1を返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

C.3.9 pjcmd_rscinfo_get_rscinfo_value()

```
pjcmd_result_t pjcmd_rscinfo_get_rscinfo_value(const PjcmdResp_t *resp_p, int indx, pjcmd_rscinfo_item_t item, void *val_p)
```

資源情報取得の応答情報に含まれる複数の資源情報から、特定の資源情報ついて値を取得します。

[引数]

resp_p

応答情報へのポインター

indx

応答情報内の参照したい資源情報のインデックス。値は、0から"関数pjcmd_rscinfo_get_rscinfo_num()が返す値-1"の範囲が指定できます。

item

資源情報内の参照したい項目の識別子(下記の表を参照)

val_p

*item*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

item	**val_p*	**val_p*の型
PJCMD_RSCINFO_ITEM_RU_NAME	リソースユニット名 **val_p*が指す領域は、応答情報解放後は不定になります。	char *
PJCMD_RSCINFO_ITEM_RU_JOB_SUBMIT	リソースユニットへのジョブの投入可否 0: ジョブの新規投入はできません。 1: ジョブの新規投入はできます。	int
PJCMD_RSCINFO_ITEM_RU_JOB_EXECUTE	リソースユニットでのジョブの実行可否 0: ジョブの新規実行はできません。 1: ジョブの新規実行はできます。	int
PJCMD_RSCINFO_ITEM_RU_SIZE	リソースユニットのサイズ ・ 計算ノードがFXサーバの場合: Tofu形状 (unsigned int)*val_p*[0] : X方向のサイズ (unsigned int)*val_p*[1] : Y方向のサイズ (unsigned int)*val_p*[2] : Z方向のサイズ ・ 計算ノードがPRIMERGYサーバの場合: ノード数 (unsigned int)*val_p*[0] : ノード数 (unsigned int)*val_p*[1] : 使用しません。 (unsigned int)*val_p*[2] : 使用しません。	unsigned intの配列 (3要素) (注)
PJCMD_RSCINFO_ITEM_RG_NAME	リソースグループ名 **val_p*が指す領域は、応答情報解放後は不定になります。	char *
PJCMD_RSCINFO_ITEM_RG_JOB_SUBMIT	リソースグループへのジョブの投入可否 0: ジョブの新規投入はできません。 1: ジョブの新規投入はできます。	int
PJCMD_RSCINFO_ITEM_RG_JOB_EXECUTE	リソースグループでのジョブの実行可否 0: ジョブの新規実行はできません。 1: ジョブの新規実行はできます。	int
PJCMD_RSCINFO_ITEM_RG_SIZE	リソースグループのサイズ ・ 計算ノードがFXサーバの場合: ノード形状 (unsigned int)*val_p*[0] : X方向のサイズ (unsigned int)*val_p*[1] : Y方向のサイズ (unsigned int)*val_p*[2] : Z方向のサイズ ・ 計算ノードがPRIMERGYサーバの場合: ノード数 (unsigned int)*val_p*[0] : ノード数 (unsigned int)*val_p*[1] : 使用しません。 (unsigned int)*val_p*[2] : 使用しません。	unsigned intの配列 (3要素) (注)
PJCMD_RSCINFO_ITEM_MAX_SIZE_NUM	ジョブに割り当て可能な最大ノード形状のバリエーションの数 関数pjcmd_rscinfo_get_max_size()で最大ノード形状を取得するときに使用します。	unsigned int

(注) 計算ノードがPRIMERGYサーバの場合は配列の先頭の要素だけ使いますが、領域は3要素分用意してください。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*item*に不明な値が指定されました。

PJCMD_ERROR_NODATA

*indx*の値が範囲外です。

C.3.10 pjcmd_rscinfo_get_max_size()

```
pjcmd_result_t pjcmd_rscinfo_get_max_size(PjcmdResp_t *resp_p, int indx1, int indx2, unsigned int *val_p)
```

応答情報に含まれる資源情報の1つについて、ジョブに割当可能な最大ノード形状を取得します。

[引数]

resp_p

応答情報へのポインター

indx1

応答情報内にある参照したい資源情報のインデックス。値は0から"関数pjcmd_rscinfo_get_rscinfo_num()で得られる値-1"の範囲が指定できます。

indx2

資源情報内にある最大ノード形状のバリエーションのうちの1つを指すインデックス。値は、0から"関数pjcmd_rscinfo_get_rscinfo_value()のパラメーターPJCMD_RSCINFO_ITEM_MAX_SIZE_NUMで得られる値-1"の範囲が指定できます。

val_p

配列*val_p*[]に最大ノード形状が格納されます。unsigned int型の配列(要素数3)を格納できる領域を呼び出し側で確保する必要があります。

- 計算ノードがFXサーバの場合: ノード形状
  val_p[0] : X方向のサイズ
  val_p[1] : Y方向のサイズ
  val_p[2] : Z方向のサイズ
- 計算ノードがPRIMERGYサーバの場合: ノード数
  val_p[0] : ノード数
  val_p[1] : 使用しません。
  val_p[2] : 使用しません。

計算ノードがPRIMERGYサーバの場合は配列の先頭の要素だけ使いますが、領域は3要素分用意してください。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*item*に不明な値が指定されました。

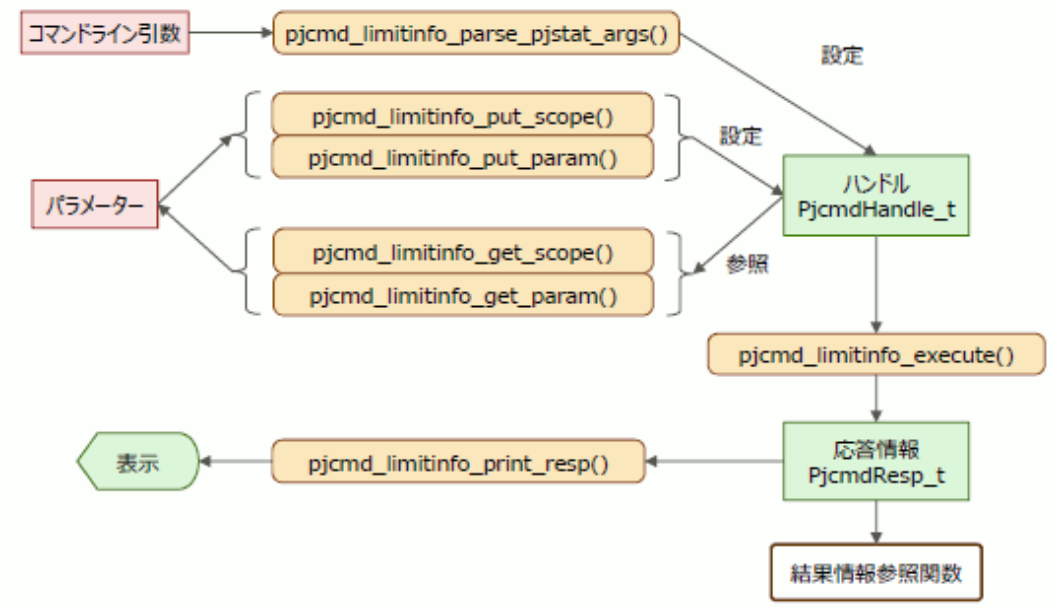
PJCMD_ERROR_NODATA

*indx1*または*indx2*が範囲外の値です。

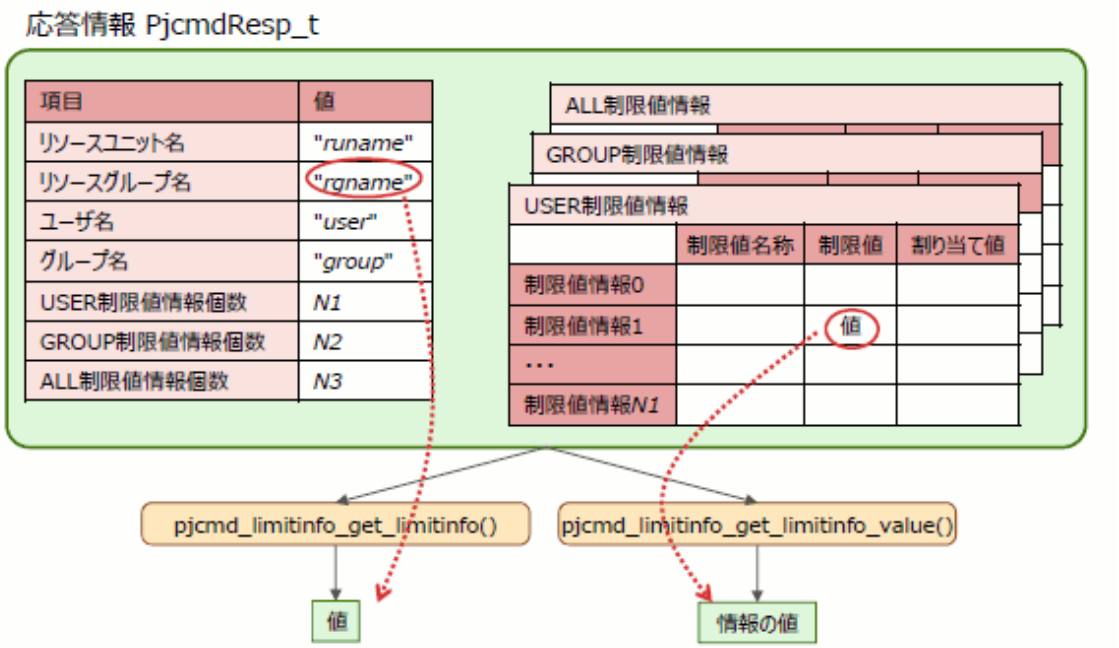
C.4 ジョブ投入時の制限値の情報取得

ここでは、ジョブ投入時の制限値についての情報を取得する関数を説明します。

図C.7 制限値情報の取得



図C.8 制限値情報の参照



C.4.1 pjcmd_limitinfo_parse_pjstat_args()

```
pjcmd_result_t pjcmd_limitinfo_parse_pjstat_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjstatコマンドの--limitオプション指定時の仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 制限値情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションでない引数が 配列*argv_pp*[]の後ろに移動します。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1] がそのオプションを指します。
なお、本関数は、pjstatコマンドの--limitオプションおよびそれと同時に指定できるオプションだけを認識します。

C.4.2 pjcmd_limitinfo_put_scope()

```
pjcmd_result_t pjcmd_limitinfo_put_scope(PjcmdHandle_t *handle_p, pjcmd_scope_t scope, void *val_p)
```

制限値情報の取得対象範囲をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

scope

情報の取得対象範囲の種類を示す識別子(下記の表を参照)

val_p

情報取得対象を示す値が格納された領域へのポインター。例えば、設定する値の型がchar *型の場合、char *型の値を格納した領域を呼び出し側で用意し、それへのポインター(char **)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

scope	**val_p*	**val_p*の型
PJCMD_SCOPE_CLUSTER	クラスタ名(1つだけ。pjstat -c相当) 設定をしない場合は、環境変数PXMYCLSTの値が適用されます。それも設定されていない場合は、関数pjcmd_limitinfo_execute()がエラーになります。 このパラメーターは、システム管理ノードで呼び出された場合のみ有効になります。システム管理ノード以外では、設定は無視され、呼び出しノードが属するクラスタが適用されます。	char *
PJCMD_SCOPE_RSCUNIT	リソースユニット名(1つだけ。pjstat --rscunit相当) 設定をしない場合は、実行ユーザーのデフォルトのリソースユニットが適用されます。	char *
PJCMD_SCOPE_RSCGRP	リソースグループ名(1つだけ。pjstat --rscgrp相当) 設定をしない場合は、リソースユニットの情報が取得されます。	char *

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 制限値情報取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値または指定できない値が指定されました。

PJCMD_ERROR_INVALID_PARAM

*val_p*が不正です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

C.4.3 pjcmd_limitinfo_get_scope()

```
pjcmd_result_t pjcmd_limitinfo_get_scope(const PjcmdHandle_t *handle_p, pjcmd_scope_t scope, void *val_p)
```

ハンドルに設定されている制限値情報取得範囲を参照します。

[引数]

handle_p

ハンドルへのポインター

scope

参照する情報取得対象範囲の識別子。指定できる識別子は、関数pjcmd_limitinfo_put_scope()と同じです。

val_p

*scope*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 制限値情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定した*scope*は、ハンドルに設定されていません。

C.4.4 pjcmd_limitinfo_put_param()

```
pjcmd_result_t pjcmd_limitinfo_put_param(PjcmdHandle_t *handle_p, pjcmd_limitinfo_param_t param, const void *val_p)
```

制限値情報取得に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

取得する制限値情報に関するパラメーターの識別子(後述の表を参照)

val_p

設定する値が格納された領域へのポインター。例えば、設定する値の型がuid_t型の場合、uid_t型の値を格納した領域を呼び出し側で用意し、それへのポインター(uid_t *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_pの型*
PJCMD_LIMITINFO_USER_ID	制限値情報取得の対象ユーザーID	uid_t
PJCMD_LIMITINFO_USER_NAME	制限値情報取得の対象ユーザー名	char *
PJCMD_LIMITINFO_GROUP_ID	制限値情報取得の対象グループID	gid_t
PJCMD_LIMITINFO_GROUP_NAME	制限値情報取得の対象グループ名	char *
PJCMD_LIMITINFO_HELP	pjstatコマンドの--help オプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、取得する情報には影響しません。	int

パラメーターPJCMD_LIMITINFO_USER_IDとPJCMD_LIMITINFO_USER_NAME、およびPJCMD_LIMITINFO_GROUP_IDとPJCMD_LIMITINFO_GROUP_NAMEは、それぞれ最後に指定されたものが有効になります。また、それらを指定しない場合は、関数を呼び出したユーザー、グループが適用されます。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 制限値情報取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法の間違い
- 間違った値

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

C.4.5 pjcmd_limitinfo_get_param()

```
pjcmd_result_t pjcmd_limitinfo_get_param(PjcmdHandle_t *handle_p, pjcmd_limitinfo_param_t param, void *val_p)
```

ハンドルに設定されている、制限値情報取得に関するパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、pjcmd_limitinfo_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 制限値情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

C.4.6 pjcmd_limitinfo_execute()

```
PjcmdResp_t *pjcmd_limitinfo_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、制限値情報の取得をジョブ運用管理機能に依頼します。

[引数]

handle_p

ハンドルへのポインター

[返り値]

制限値情報取得の応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。情報の取得の依頼が失敗した場合は、NULLを返し、pjcmd_errcodeが設定されます。

なお、応答情報は依頼が成功したか否かを示します。情報が正常に取得できたかどうかは、関数pjcmd_get_result()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 制限値情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノード、計算クラスタ管理ノード、およびシステム管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_INTERNAL

内部エラー

C.4.7 pjcmd_limitinfo_print_resp()

```
pjcmd_result_t pjcmd_limitinfo_print_resp(const PjcmdResp_t *resp_p)
```

制限情報の取得結果を、pjstatコマンドの--limitオプション指定時の仕様に従って標準出力に出力します。

[引数]

resp_p

応答情報へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULL

- 制限値情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

C.4.8 pjcmd_limitinfo_get_limitinfo()

```
pjcmd_result_t pjcmd_limitinfo_get_limitinfo(const PjcmdResp_t *resp_p, pjcmd_limitinfo_item_t item, void *val_p)
```

制限値情報取得結果から、特定の項目の情報を取得します。

[引数]

resp_p

応答情報へのポインター

item

取得する情報項目を示す識別子(下記の表を参照)

val_p

*item*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

item	**val_p*	**val_p*の型
PJCMD_LIMITINFO_ITEM_RSCUNIT	リソースユニット名	char *
PJCMD_LIMITINFO_ITEM_RSCGRP	リソースグループ名	char *
PJCMD_LIMITINFO_ITEM_USER	対象ユーザー名	char *
PJCMD_LIMITINFO_ITEM_GROUP	対象グループ名	char *
PJCMD_LIMITINFO_ITEM_USER_INFO_NUM	ユーザーに対する制限値情報の個数	uint32_t
PJCMD_LIMITINFO_ITEM_GROUP_INFO_NUM	グループに対する制限値情報の個数	uint32_t
PJCMD_LIMITINFO_ITEM_ALL_INFO_NUM	全ユーザーの合計値に対する制限値情報の個数	uint32_t

リソースユニット名、リソースグループ名、対象ユーザー名、および対象グループ名は、応答情報内に格納された文字列へのポインターです。このため、応答情報を解放後にこれらの文字列を参照した場合の動作は不定です。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 制限値情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*item*に不明な値が指定されました。

C.4.9 pjcmd_limitinfo_get_limitinfo_value()

```
pjcmd_result_t pjcmd_limitinfo_get_limitinfo_value(const PjcmdResp_t *resp_p, pjcmd_limitinfo_class_t class, int indx,
pjcmd_limitinfo_value_t item, void *val_p)
```

制限値情報取得の応答情報から、ユーザー制限情報、グループ制限情報、または全体制限情報の中の特定の情報の値を取得します。

[引数]

resp_p

応答情報へのポインター

class

参照する制限値情報の階層(ユーザー制限情報、グループ制限情報、または全体制限情報)を示す識別子。

class	説明
PJCMD_LIMITINFO_CLASS_USER	ユーザーに対する制限値情報
PJCMD_LIMITINFO_CLASS_GROUP	グループに対する制限値情報
PJCMD_LIMITINFO_CLASS_ALL	全ユーザーの合計値に対する制限値情報

indx

*class*で示された制限値情報の階層における参照情報のインデックス。値は、0から"関数pjcmd_limitinfo_get_limitinfo()で取得するそれぞれの階層の制限値情報の個数-1"の範囲が指定できます。

item

参照する制限値情報の項目を示す識別子(下記の表を参照)。

val_p

*item*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

item	**val_p*	**val_p*の型
PJCMD_LIMITINFO_VALUE_NAME	制限値の名称	char *
PJCMD_LIMITINFO_VALUE_UPPER	上限値	uint64_t
PJCMD_LIMITINFO_VALUE_ALLOC	割り当て量	uint64_t

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 制限値情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*item*に不明な値が指定されました。

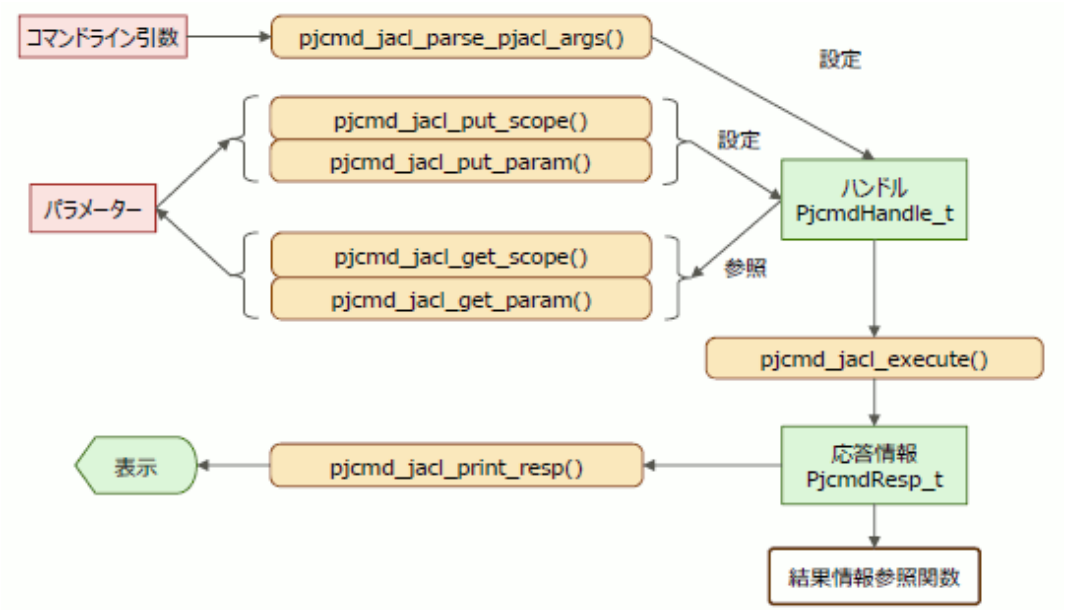
PJCMD_ERROR_NODATA

*indx*の値が範囲外です。

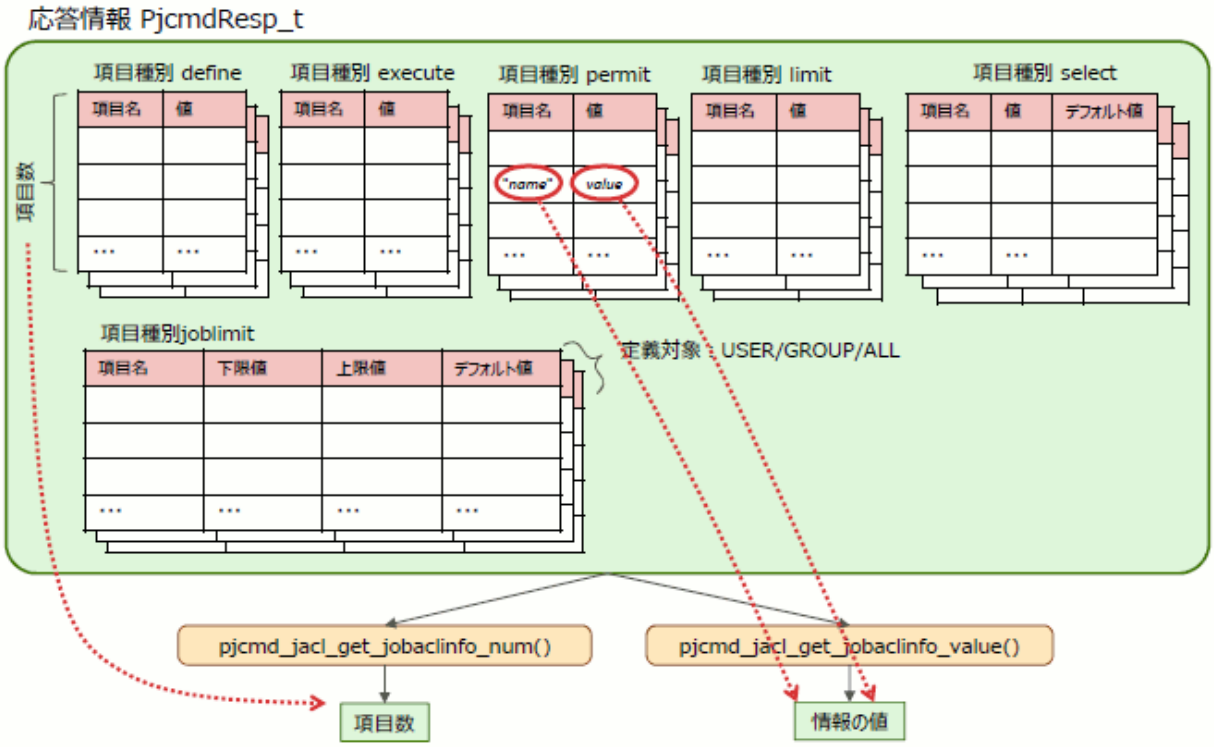
C.5 ジョブACL機能の設定情報の取得

ここでは、ジョブACL機能の設定情報を参照するための関数を説明します。

図C.9 ジョブACL機能の設定情報の取得依頼



図C.10 ジョブACL機能の設定情報の参照



C.5.1 pjcmd_jacl_parse_pjacl_args()

```
pjcmd_result_t pjcmd_jacl_parse_pjacl_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjaclコマンドのオプション仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブACL情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションでない引数が 配列*argv_pp*[]の後ろに移動します。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1] がそのオプションを指します。

C.5.2 pjcmd_jacl_put_scope()

```
pjcmd_result_t pjcmd_jacl_put_scope(PjcmdHandle_t *handle_p, pjcmd_scope_t scope, void *val_p)
```

ジョブACL情報の取得範囲をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

scope

情報の取得対象範囲の種類を示す識別子(下記の表を参照)

val_p

情報取得対象を示す値が格納された領域へのポインター。例えば、設定する値の型がchar *型の場合、char *型の値を格納した領域を呼び出し側で用意し、それへのポインター(char **)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

scope	**val_p*	**val_p*の型
PJCMD_SCOPE_RSCUNIT	リソースユニット名(1つだけ。pjacl --rscunit相当) 設定をしない場合は、実行ユーザーのデフォルトのリソースユニットが適用されます。	char *
PJCMD_SCOPE_RSCGRP	リソースグループ名(1つだけ。pjacl --rscgrp相当) 設定をしない場合は、リソースユニットの情報が取得されます。	char *

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブACL情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_PARAM

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値または指定できない値が指定されました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

C.5.3 pjcmd_jacl_get_scope()

```
pjcmd_result_t pjcmd_jacl_get_scope(const PjcmdHandle_t *handle_p, pjcmd_scope_t scope, void *val_p)
```

ハンドルに設定されているジョブACL情報の取得対象範囲名(リソースユニット、リソースグループ)を参照します。

[引数]

handle_p

ハンドルへのポインター

scope

参照する情報取得対象範囲の識別子。指定できる識別子は、関数pjcmd_jacl_put_scope()と同じです。

val_p

*scope*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULL
- ジョブACL情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定した*scope*は、ハンドルに設定されていません。

C.5.4 pjcmd_jacl_put_param()

```
pjcmd_result_t pjcmd_jacl_put_param(PjcmdHandle_t *handle_p, pjcmd_jacl_param_t param, void *val_p)
```

ジョブACL情報取得に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

取得するジョブACL情報に関するパラメーターの識別子(下記の表を参照)

val_p

設定する値が格納された領域へのポインター。例えば、設定する値の型がuid_t型の場合、uid_t型の値を格納した領域を呼び出し側で用意し、それへのポインター(uid_t *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_JACL_USER_ID	制限値情報取得の対象ユーザーID	uid_t
PJCMD_JACL_USER_NAME	制限値情報取得の対象ユーザー名	char *
PJCMD_JACL_GROUP_ID	制限値情報取得の対象グループID	gid_t
PJCMD_JACL_GROUP_NAME	制限値情報取得の対象グループ名	char *
PJCMD_JACL_DATA	pjaclコマンドの--dataオプションの指定に相当。 0 : 指定なし(デフォルト)。 1 : 指定あり。	int

param	**val_p*	**val_pの型*
	このパラメーターは、取得する情報には影響しませんが、関数pjcmd_jacl_print_resp()による出力結果に影響します。	
PJCMD_JACL_DELIMITER	pjaclコマンドの--delimiterオプションによる表示区切り文字の指定に相当。 このパラメーターは、取得する情報には影響しませんが、関数pjcmd_jacl_print_resp()による出力結果に影響します。このパラメーターを指定しない場合は、コンマ(,)が使用されます。	char *
PJCMD_JACL_HELP	pjaclコマンドの--helpオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、取得するジョブ情報には影響しません。	int

パラメーター PJCMD_JACL_USER_ID と PJCMD_JACL_USER_NAME 、および PJCMD_JACL_GROUP_ID と PJCMD_JACL_GROUP_NAMEは、それぞれ最後に指定されたものが有効になります。また、それらを指定しない場合は、関数を呼び出したユーザー、グループが適用されます。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブACL情報取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

*val_p*が不正です。

- 指定方法の間違い
- 間違った値

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

C.5.5 pjcmd_jacl_get_param()

```
pjcmd_result_t pjcmd_jacl_get_param(const PjcmdHandle_t *handle_p, pjcmd_jacl_param_t param, void *val_p)
```

ハンドルに設定されている、ジョブACL情報取得に関するパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、pjcmd_jacl_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブACL情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

C.5.6 pjcmd_jacl_execute()

```
PjcmdResp_t *pjcmd_jacl_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、ジョブACL情報の取得をジョブ運用管理機能に依頼します。

[引数]

handle_p

ハンドルへのポインター

[返り値]

ジョブACL情報取得の応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。情報の取得の依頼が失敗した場合は、NULLを返し、pjcmd_errcodeが設定されます。
なお、応答情報は依頼が成功したか否かを示します。情報が正常に取得できたかどうかは、関数pjcmd_get_result()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULL
- ジョブACL情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この本関数は、ログインノードと計算クラスタ管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、情報取得依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_INTERNAL

内部エラー

C.5.7 pjcmd_jacl_print_resp()

```
pjcmd_result_t pjcmd_jacl_print_resp(const PjcmdResp_t *resp_p)
```

ジョブACL情報の取得結果を、pjaclコマンドの仕様に従って標準出力に出力します。

[引数]

resp_p

応答情報へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- ジョブACL情報取得の応答情報ではありません。
- 情報取得が成功した応答情報ではありません。

C.5.8 pjcmd_jacl_get_jaclinfo_num()

```
int pjcmd_jacl_get_jaclinfo_num(const PjcmdResp_t *resp_p, pjcmd_jacl_class_t class, pjcmd_jacl_type_t type)
```

ジョブACL情報取得の応答情報から、特定の定義対象および特定のジョブACL定義種別について、ジョブACL定義項目の総数を取得します。

[引数]

resp_p

応答情報へのポインター

class

取得するジョブACL定義項目の定義対象(USER、GROUP、またはALL定義)を示す識別子

class	説明
PJCMD_JACL_CLASS_USER	USER定義
PJCMD_JACL_CLASS_GROUP	GROUP定義
PJCMD_JACL_CLASS_ALL	ALL定義

type

取得するジョブACL定義項目の定義種別(define、joblimit、execute、permit、limit、またはselect)を示す識別子

type	説明
PJCMD_JACL_TYPE_DEFINE	定義種別define
PJCMD_JACL_TYPE_JOBLIMIT	定義種別joblimit
PJCMD_JACL_TYPE_EXECUTE	定義種別execute
PJCMD_JACL_TYPE_PERMIT	定義種別permit
PJCMD_JACL_TYPE_LIMIT	定義種別limit
PJCMD_JACL_TYPE_SELECT	定義種別select

[返り値]

ジョブACL定義項目の総数。
失敗した場合は-1を返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULL
- ジョブACL情報取得の応答情報ではありません。
- 情報取得が成功した応答情報ではありません。

PJCMD_ERROR_UNKNOWN_PARAM

不明な*class*または*type*が指定されました。

PJCMD_ERROR_INVALID_PARAM

*class*と*type*の組み合わせで示される定義項目はありません。

C.5.9 pjcmd_jacl_get_jaclinfo_value()

```
pjcmd_result_t pjcmd_jacl_get_jaclinfo_value(const PjcmdResp_t *resp_p, pjcmd_jacl_class_t class, pjcmd_jacl_type_t type,
int indx, pjcmd_jacl_info_t info, void *val_p)
```

ジョブACL定義情報の値を参照します。

[引数]

resp_p

応答情報へのポインター

class

ジョブACL定義項目の定義対象(USER、GROUP、またはALL定義)を示す識別子。指定できる値は関数pjcmd_jacl_get_jaclinfo_num()と同じです。

type

ジョブACL定義項目の定義種別(define、joblimit、execute、permit、limit、またはselect)を示す識別子。指定できる値は関数pjcmd_jacl_get_jaclinfo_num()と同じです。

indx

定義対象*class*、定義種別*type*のジョブACL定義項目における、参照したい定義項目のインデックス。値は、0から"関数pjcmd_jacl_get_jaclinfo_num()で得られる値-1"の範囲が指定できます。

info

参照する値の識別子(下記の表を参照)。

val_p

*info*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

info	**val_p*	**val_p*の型
PJCMD_JACL_INFO_NAME	ジョブACL定義項目名 ジョブACL定義を設定する際に記述する定義項目名と同じです。	char *
PJCMD_JACL_INFO_DEFINE_VALUE	定義種別defineのジョブACL定義項目の値	char *
PJCMD_JACL_INFO_JOBLIMIT_LOWER	定義種別joblimitのジョブACL定義項目の下限値	uint64_t
PJCMD_JACL_INFO_JOBLIMIT_UPPER	定義種別joblimitのジョブACL定義項目の上限値	uint64_t
PJCMD_JACL_INFO_JOBLIMIT_DEFAULT	定義種別joblimitのジョブACL定義項目の初期値	uint64_t
PJCMD_JACL_INFO_EXECUTE_VALUE	定義種別executeのジョブACL定義項目の値	char *
PJCMD_JACL_INFO_PERMIT_VALUE	定義種別permitのジョブACL定義項目の値	char *
PJCMD_JACL_INFO_LIMIT_VALUE	定義種別limitのジョブACL定義項目の値	uint64_t
PJCMD_JACL_INFO_SELECT_VALUE	定義種別selectのジョブACL定義項目の値	char *
PJCMD_JACL_INFO_SELECT_DEFAULT	定義種別selectのジョブACL定義項目の初期値	char *

値の型がchar *型(文字列へのポインター)の場合、文字列は応答情報内の領域にあります。このため、応答情報を解放後に文字列を参照すると、動作は不定です。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- ジョブACL情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

class、*type*または*info*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

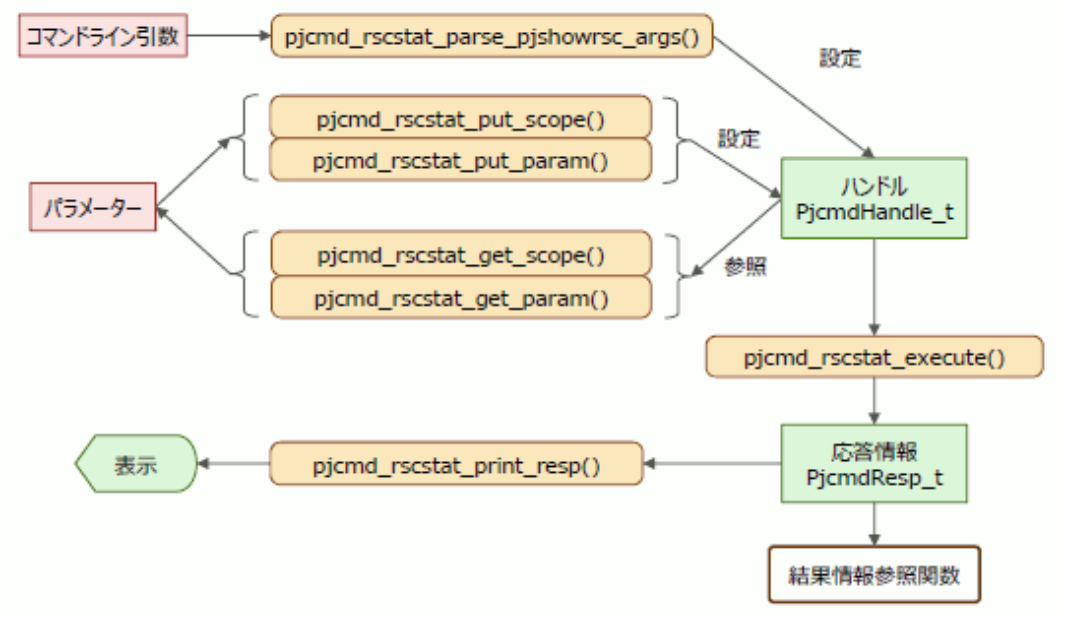
指定されたパラメーターが不正です。

- ジョブACL定義項目*type*には情報*info*は指定できません。
- *class*と*type*の組み合わせで示される定義項目はありません。
- *indx*の値が範囲外です。

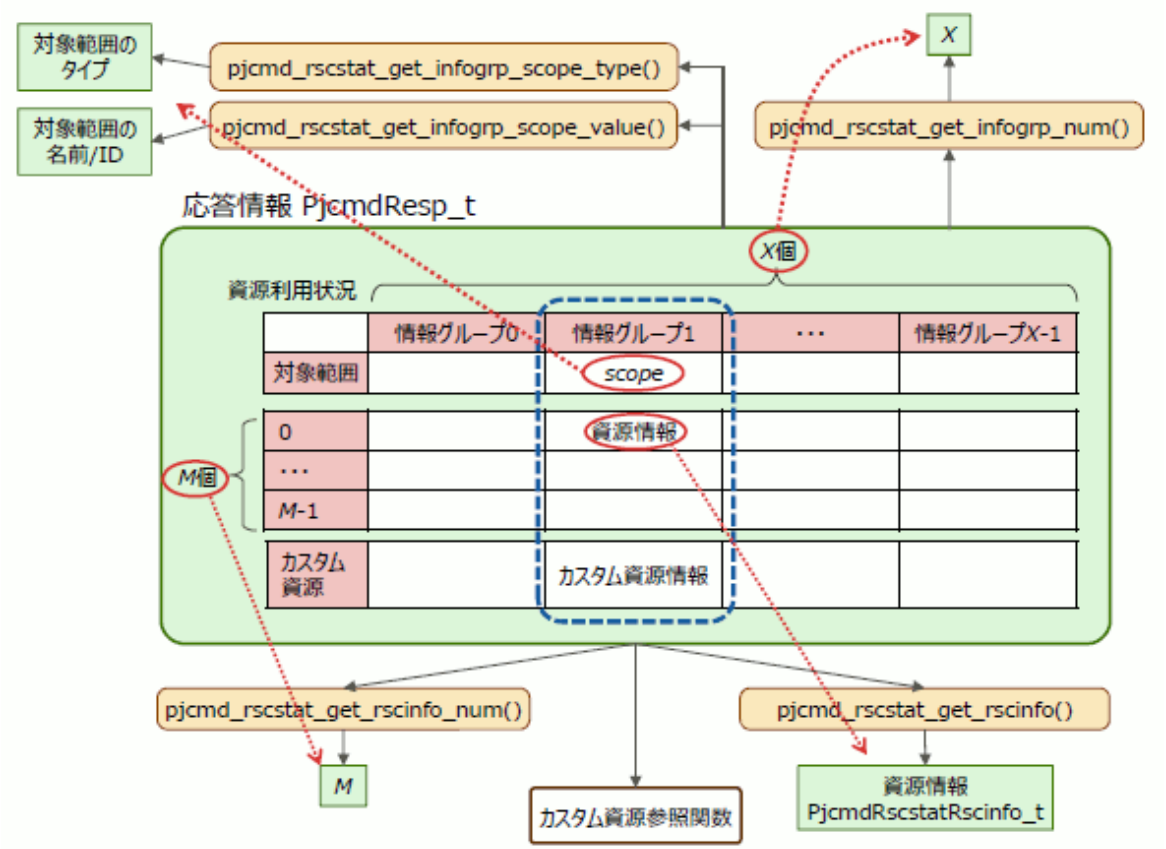
C.6 ジョブ用資源の利用状況の取得

ここでは、ジョブ用資源の利用状況を取得するための関数を説明します。

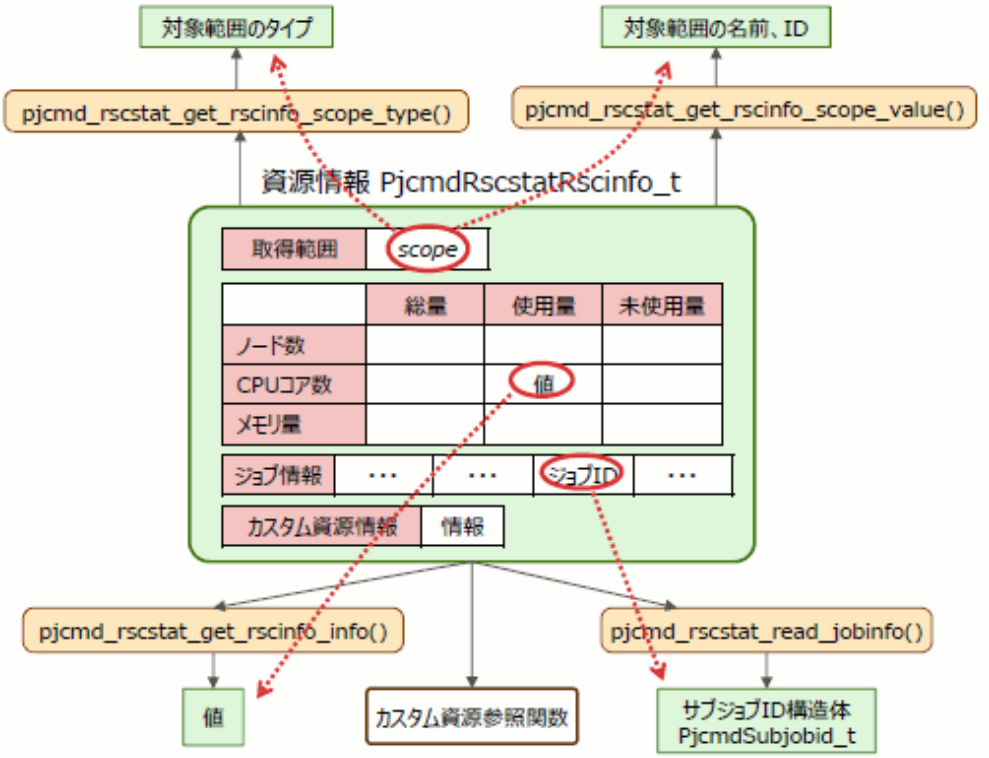
図C.11 ジョブ用資源の利用状況の取得依頼



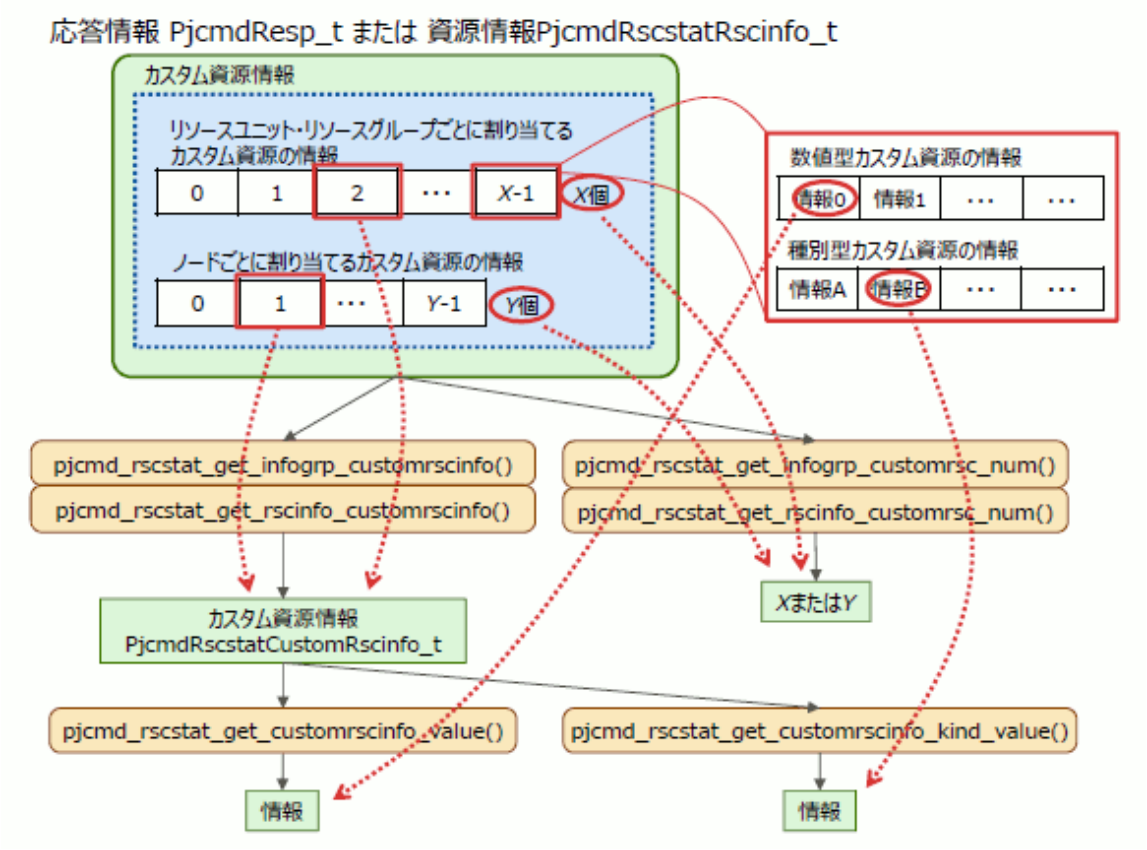
図C.12 情報グループ(情報の取得単位)の参照



図C.13 資源情報の参照



図C.14 カスタム資源情報の参照



C.6.1 pjcmd_rscstat_parse_pjshowrsc_args()

```
pjcmd_result_t pjcmd_rscstat_parse_pjshowrsc_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpjshowrscコマンドのオプション仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。

- 資源の利用状況取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。

PJCMD_ERROR_NOMEM

メモリの獲得が失敗しました。

PJCMD_ERROR_INTERNAL

内部エラー

本関数の呼び出しにより、オプションでない引数が 配列*argv_pp*[]の後ろに移動します。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1] がそのオプションを指します。

C.6.2 pjcmd_rscstat_put_scope()

```
pjcmd_result_t pjcmd_rscstat_put_scope(PjcmdHandle_t *handle_p, pjcmd_scope_t scope, const void *val_p, uint32_t n)
```

資源の利用状況の取得範囲をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

scope

情報の取得対象範囲の種類を示す識別子(下記の表を参照)

val_p

情報取得対象を示す値が格納された領域へのポインター。例えば、設定する値の型がchar *型の場合、char *型の値を格納した領域を呼び出し側で用意し、それへのポインター(char **)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

n

*val_p*の要素数

scope	**val_p*	**val_p*の型
PJCMD_SCOPE_CLUSTER	クラスタ名(1つだけ。pjshowrsc -c相当) 設定をしない場合は、環境変数PXMYCLSTの値が適用されます。それも設定されていない場合は、権限があるすべてのクラスタが対象になります。 このパラメーターは、システム管理ノードで呼び出された場合のみ有効になります。システム管理ノード以外では、設定は無視され、呼び出しノードが属するクラスタが適用されます。 また、このパラメーターに対しては、*n*には1を指定してください。	char *
PJCMD_SCOPE_NODEGRP	ノードグループIDの配列(要素数*n*。pjshowrsc --nodegrp相当) このパラメーターをハンドルに設定した場合、関数pjcmd_rscstat_execute()の呼び出しには管理者権限が必要です。	uint32_t *

scope	**val_p*	**val_p*の型
PJCMD_SCOPE_NODEGRP_STR	ノードグループID(文字列)の配列(要素数*n*。pjshowrsc --nodegrp相当) ノードグループIDは、範囲表現("*ID1-ID2*")も使用できます。このパラメーターをハンドルに設定した場合、関数pjcmd_rscstat_execute()の呼び出しには管理者権限が必要です。	char **
PJCMD_SCOPE_BOOTGRP	ブートグループIDの配列(要素数*n*。pjshowrsc --bootgrp相当) このパラメーターをハンドルに設定した場合、関数pjcmd_rscstat_execute()の呼び出しには管理者権限が必要です。	uint32_t *
PJCMD_SCOPE_BOOTGRP_STR	ブートグループID(文字列)の配列(要素数*n*。pjshowrsc --bootgrp相当) ブートグループIDは、範囲表現("*ID1-ID2*")も使用できます。このパラメーターをハンドルに設定した場合、関数pjcmd_rscstat_execute()の呼び出しには管理者権限が必要です。	char **
PJCMD_SCOPE_RSCUNIT	リソースユニット名の配列(要素数*n*。pjshowrsc --rscunit相当) リソースユニット名として"*"を指定した場合は、すべてのリソースユニットが対象になります。	char **
PJCMD_SCOPE_RSCGRP	リソースグループ名の配列(要素数*n*。pjshowrsc --rscgrp相当) リソースグループ名として"*"を指定した場合は、すべてのリソースグループが対象になります。	char **
PJCMD_SCOPE_NODE	ノードIDの配列(要素数*n*。pjshowrsc -n相当)	uint32_t *
PJCMD_SCOPE_NODE_STR	ノードID(文字列)の配列(要素数*n*。pjshowrsc -n相当) ノードIDは、範囲表現("*ID1-ID2*")も使用できます。	char **

PJCMD_SCOPE_CLUSTER以外の*scope*は、互いに排他です。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 資源の利用状況取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

*val_p*または*n*が不正です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

C.6.3 pjcmd_rscstat_get_scope()

```
pjcmd_result_t pjcmd_rscstat_get_scope(const PjcmdHandle_t *handle_p, pjcmd_scope_t scope, void *val_p, uint32_t *n_p)
```

ハンドルに設定されている資源状態の取得対象範囲を参照します。

[引数]

handle_p

ハンドルへのポインター

scope

参照する情報取得対象範囲の識別子。指定できる識別子は、関数pjcmd_rscstat_put_scope()と同じです。

val_p

*scope*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

n_p

**n_p*に*val_p*の要素数が格納されます。領域は呼び出し側で用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 資源の利用状況取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*または*n*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定した*scope*は、ハンドルに設定されていません。

C.6.4 pjcmd_rscstat_put_param()

```
pjcmd_result_t pjcmd_rscstat_put_param(PjcmdHandle_t *handle_p, pjcmd_rscstat_param_t param, const void *val_p)
```

資源状態の取得に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

取得する資源の利用状況に関するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値(**val_p*)の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_RSCSTAT_VERBOSITY	取得する資源状態情報の粒度 ・ PJCMD_RSCSTAT_VERBOSITY_SELF 指定した*scope*の階層の情報を取得します(デフォルト)。 ・ PJCMD_RSCSTAT_VERBOSITY_CHILD 指定した*scope*の下の階層も取得します。 ・ PJCMD_RSCSTAT_VERBOSITY_NODE ノードの資源状態まで取得します。	int
PJCMD_RSCSTAT_INFO_LEVEL	パラメーターPJCMD_RSCSTAT_VERBOSITYで、ノード単位の情報取得(PJCMD_RSCSTAT_VERBOSITY_NODE)を指定した場合に取得する情報のレベル(pjshowrsc -v相当)。 0: 計算機資源量(ノード数、CPUコア数、メモリ量、ローカルファイルシステムサイズ)を取得します(デフォルト)。 1: 上記の情報に加えて、実行中のジョブIDも取得します。 2: 上記の情報に加えて、カスタム資源情報も取得します。 3: 上記の情報に加えて、ノードを通信経路として利用しているジョブIDも取得します。	int
PJCMD_RSCSTAT_EXCLUSIVE	取得するリソースグループの資源使用量から、資源を共有するほかのリソースグループ内のジョブについて除外するか否かを指定します(pjshowrsc --exclusive相当)。 0 : ほかのリソースグループの資源を含めます(デフォルト)。 1 : ほかのリソースグループの資源は除外します。	int
PJCMD_RSCSTAT_CUSTOMRSC	リソースユニットまたはリソースグループが対象の場合に、カスタム資源情報を取得するか否かを指定します(pjshowrsc --custom-resource相当)。 0 : カスタム資源情報を取得しません(デフォルト)。 1 : カスタム資源情報を取得します。	int
PJCMD_RSCSTAT_STATUS	ノードが対象の場合に、すべてのノードについて情報を取得するか否かを指定します。 0 : 利用可能なノードについてだけ取得します(デフォルト)。 1 : すべてのノードについて取得します。	int
PJCMD_RSCSTAT_RAW	pjshowrscコマンドの--rawオプションの指定に相当。 0 : 指定なし(デフォルト)。 1 : 指定あり。 このパラメーターは、取得する情報には影響しませんが、関数pjcmd_rscstat_print_resp()による出力結果に影響します。	int
PJCMD_RSCSTAT_DATA	pjshowrscコマンドの--dataオプションの指定に相当。 0 : 指定なし(デフォルト)。 1 : 指定あり。 このパラメーターは、取得する情報には影響しませんが、関数pjcmd_rscstat_print_resp()による出力結果に影響します。	int
PJCMD_RSCSTAT_DELIMITER	pjshowrscコマンドの--delimiterオプションによる表示の区切り文字の指定に相当。	char *

param	**val_p*	**val_pの型*
	このパラメーターは、取得する情報には影響しませんが、関数pjcmd_rscstat_print_resp()による出力結果に影響します。このパラメーターを指定しない場合は、コンマ(,)が使用されます。	
PJCMD_RSCSTAT_HELP	pjshowrscコマンドの--helpオプションの指定に相当。 0: 指定なし(デフォルト)。 1: 指定あり。 このパラメーターは、取得する情報には影響しません。	int

ジョブ資源利用について取得する情報は、関数pjcmd_rscstat_put_scope()で指定する*scope*と、関数pjcmd_rscstat_put_param()のパラメーターPJCMD_RSCSTAT_VERBOSITYで指定する情報の粒度によって、以下の単位で取得されます。

scope	粒度(*)	得られる情報の単位
クラスタ	SELF	クラスタ単位
	CHILD	計算クラスタサブ管理ノードがあるシステムの場合: ノードグループ 計算クラスタサブ管理ノードがないシステムの場合: ブートグループ
	NODE	ノード
ノードグループ	SELF	ノードグループ
	CHILD	ブートグループ
	NODE	ノード
ブートグループ	SELF	ブートグループ
	CHILD	ノード
	NODE	ノード
リソースユニット	SELF	リソースユニット
	CHILD	ノード
	NODE	ノード
リソースグループ	SELF	リソースグループ
	CHILD	ノード
	NODE	ノード
ノード	任意	ノード

(*)"SELF" は PJCMD_RSCSTAT_VERBOSITY_SELF 、 "CHILD" は PJCMD_RSCSTAT_VERBOSITY_CHILD 、 および"NODE"はPJCMD_RSCSTAT_VERBOSITY_NODEを意味します。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。

- 資源の利用状況取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法の間違い
- 間違った値

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

C.6.5 pjcmd_rscstat_get_param()

```
pjcmd_result_t pjcmd_rscstat_get_param(const PjcmdHandle_t *handle_p, pjcmd_rscstat_param_t param, void *val_p)
```

ハンドルに設定されている、資源状態の取得に関するパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、pjcmd_rscstat_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- 資源の利用状況取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

C.6.6 pjcmd_rscstat_execute()

```
PjcmdResp_t *pjcmd_rscstat_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、資源の利用状況の取得をジョブ運用管理機能に依頼します。

[引数]

handle_p

ハンドルへのポインター

[返り値]

資源の利用状況の取得の応答情報。
得られた応答情報は、関数pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。情報の取得の依頼が失敗した場合は、NULLを返し、pjcmd_errcodeが設定されます。
なお、応答情報は依頼が成功したか否かを示します。情報が正常に取得できたかどうかは、関数pjcmd_get_result()で応答情報から結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULL
- 資源の利用状況取得用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。
この関数は、ログインノード、計算クラスタ管理ノード、およびシステム管理ノードで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_CONNECT

ジョブ運用管理機能のデーモンとの通信に失敗しました。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_BUSY

ほかの操作依頼関数が処理中のため、操作依頼はできません。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_SIGNAL

シグナルを受信したため処理を中断しました。

PJCMD_ERROR_INTERNAL

内部エラー

C.6.7 pjcmd_rscstat_print_resp()

```
pjcmd_result_t pjcmd_rscstat_print_resp(const PjcmdResp_t *resp_p)
```

資源の利用状況の取得結果を、pjshowrscコマンドの仕様に従って標準出力に出力します。

[引数]

resp_p

応答情報へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源の利用状況取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

C.6.8 pjcmd_rscstat_get_infogrp_num()

```
int pjcmd_rscstat_get_infogrp_num(const PjcmdResp_t *resp_p)
```

資源の利用状況取得の応答情報に含まれる情報グループの数を取得します。
情報グループは、情報取得時に関数pjcmd_rscstat_put_scope()で指定した情報の取得単位(クラスタ、ノードグループ、ブートグループなど)ごとの情報です。

[引数]

resp_p

応答情報へのポインター

[返り値]

情報グループの数。
失敗した場合は-1を返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源の利用状況取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

C.6.9 pjcmd_rscstat_get_infogrp_scope_type()

```
pjcmd_scope_t pjcmd_rscstat_get_infogrp_scope_type(const PjcmdResp_t *resp_p, int indx)
```

資源の利用状況取得の応答情報に含まれる特定の情報グループについて、情報取得単位のタイプを返します。

[引数]

resp_p

応答情報へのポインター

indx

情報グループのインデックス。値は、0から"関数pjcmd_rscstat_get_infogrp_num()で得られる値-1"の範囲が指定できます。

[返り値]

情報グループの情報取得単位を示す識別子。

識別子	意味
PJCMD_SCOPE_CLUSTER	クラスタ単位
PJCMD_SCOPE_NODEGRP	ノードグループ単位
PJCMD_SCOPE_BOOTGRP	ブートグループ単位
PJCMD_SCOPE_RSCUNIT	リソースユニット単位
PJCMD_SCOPE_RSCGRP	リソースグループ単位
PJCMD_SCOPE_NODE	ノード単位

失敗した場合は、-1を返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源の利用状況取得の応答情報ではありません。
- 情報取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_PARAM

*indx*の値が範囲外です。

C.6.10 pjcmd_rscstat_get_infogrp_scope_value()

```
pjcmd_result_t pjcmd_rscstat_get_infogrp_scope_value(const PjcmdResp_t *resp_p, int indx, pjcmd_scope_t scope, void *val_p)
```

資源の利用状況取得の応答情報に含まれる特定の情報グループについて、情報取得単位の名称(クラスタ名、リソースユニット名、またはリソースグループ名)またはID(ノードグループID、ブートグループID、またはノードID)を取得します。

[引数]

resp_p

応答情報へのポインター

indx

情報グループのインデックス。値は、0から"関数pjcmd_rscstat_get_infogrp_num()で得られる値-1"の範囲が指定できます。

scope

名称またはIDを取得する情報取得単位のタイプ(下記の表を参照)。
引数*scope*が情報グループの取得単位のタイプと異なる場合はエラーになります。

val_p

*scope*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を呼び出す側が用意する必要があります。

scope	**val_p*	**val_p*の型
PJCMD_SCOPE_CLUSTER	クラスタ名 **val_p*が指す領域は、応答情報解放後は不定になります。	char *
PJCMD_SCOPE_NODEGRP	ノードグループID	uint32_t
PJCMD_SCOPE_BOOTGRP	ブートグループID	uint32_t
PJCMD_SCOPE_RSCUNIT	リソースユニット名 **val_p*が指す領域は、応答情報解放後は不定になります。	char *
PJCMD_SCOPE_RSCGRP	リソースグループ名 **val_p*が指す領域は、応答情報解放後は不定になります。	char *

scope	**val_p*	**val_pの型*
PJCMD_SCOPE_NODE	ノードID	uint32_t

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源の利用状況取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

- *scope*は、情報グループの取得単位と一致しません。
- *indx*の値が範囲外です。

C.6.11 pjcmd_rscstat_get_rscinfo_num()

```
int pjcmd_rscstat_get_rscinfo_num(const PjcmdResp_t *resp_p, int indx)
```

資源の利用状況取得の応答情報における、特定の情報グループに含まれる資源情報の数を取得します。

[引数]

resp_p

応答情報へのポインター

indx

情報グループのインデックス。値は、0から"関数pjcmd_rscstat_get_infogrp_num()で得られる値-1"の範囲が指定できます。

[返り値]

資源情報の数。
失敗した場合は-1を返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源の利用状況取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_PARAM

*indx*の値が範囲外です。

C.6.12 pjcmd_rscstat_get_infogrp_customrsc_num()

```
pjcmd_result_t pjcmd_rscstat_get_infogrp_customrsc_num(const PjcmdResp_t *resp_p, int indx,
pjcmd_rscstat_customrsc_alloc_t type, uint32_t *num_p)
```

特定の情報グループについて、カスタム資源の数を取得します。

[引数]

resp_p

応答情報へのポインター

indx

情報グループのインデックス。値は、0から"関数pjcmd_rscstat_get_infogrp_num()で得られる値-1"の範囲が指定できます。

type

カスタム資源の割当タイプ

識別子	意味
PJCMD_RSCSTAT_CUSTOMRSC_ALLOC_RU_RG	リソースユニットまたはリソースグループごとに割り当てるカスタム資源
PJCMD_RSCSTAT_CUSTOMRSC_ALLOC_NODE	ノードごとに割り当てるカスタム資源

num_p

**num_p*にカスタム資源の数が格納されます。領域は呼び出し側で用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源の利用状況取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_PARAM

*indx*の値が範囲外です。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*type*に不明な値が指定されました。

C.6.13 pjcmd_rscstat_get_infogrp_customrscinfo()

```
PjcmdRscstatCustomRscinfo_t * pjcmd_rscstat_get_infogrp_customrscinfo(const PjcmdResp_t *resp_p, int indx,
pjcmd_rscstat_customrsc_alloc_t type, uint32_t cs_indx)
```

特定の情報グループについて、カスタム資源の情報を取得します。

[引数]

resp_p

応答情報へのポインター

indx

情報グループのインデックス。値は、0から"関数pjcmd_rscstat_get_infogrp_num()で得られる値-1"の範囲が指定できます。

type

カスタム資源の割当タイプ

識別子	意味
PJCMD_RSCSTAT_CUSTOMRSC_ALLOC_RU_RG	リソースユニットまたはリソースグループごとに割り当てるカスタム資源
PJCMD_RSCSTAT_CUSTOMRSC_ALLOC_NODE	ノードごとに割り当てるカスタム資源

cs_indx

カスタム資源のインデックス。値は、0から"関数pjcmd_rscstat_get_infogrp_customrsc_num()で得られる値-1"の範囲が指定できます。

[返り値]

カスタム資源情報へのポインター。得られる情報は、応答情報の解放後に参照した場合の動作は不定です。
失敗時はNULLを返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源の利用状況取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_PARAM

*indx*の値が範囲外です。

PJCMD_ERROR_UNKNOWN_PARAM

*type*に不明な値が指定されました。

PJCMD_ERROR_NODATA

*cs_indx*の値が範囲外です。

C.6.14 pjcmd_rscstat_get_rscinfo()

```
PjcmdRscstatRscinfo_t *pjcmd_rscstat_get_rscinfo(const PjcmdResp_t *resp_p, int infogrp_indx, int rscinfo_indx)
```

特定の情報グループ内の1つの資源情報を取得します。

[引数]

resp_p

応答情報へのポインター

infogrp_indx

情報グループのインデックス。値は、0から"関数pjcmd_rscstat_get_infogrp_num()で得られる値-1"の範囲が指定できます。

rscinfo_indx

情報グループ内の参照する資源情報のインデックス。値は、0から"関数pjcmd_rscstat_get_rscinfo_num()で得られる値-1"の範囲が指定できます。

[返り値]

資源情報へのポインター。
失敗した場合はNULLを返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- 資源の利用状況取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

PJCMD_ERROR_INVALID_PARAM

*infogrp_indx*または*rscinfo_indx*の値が範囲外です。

C.6.15 pjcmd_rscstat_get_rscinfo_scope_type()

```
pjcmd_scope_t pjcmd_rscstat_get_rscinfo_scope_type(const PjcmdRscstatRscinfo_t *rscinfo_p)
```

資源情報について、情報取得単位のタイプを取得します。

[引数]

rscinfo_p

資源情報へのポインター。関数pjcmd_rscstat_get_rscinfo()が返す値です。

[返り値]

資源情報の情報取得単位を示す識別子

識別子	意味
PJCMD_SCOPE_CLUSTER	クラスタ単位
PJCMD_SCOPE_NODEGRP	ノードグループ単位
PJCMD_SCOPE_BOOTGRP	ブートグループ単位
PJCMD_SCOPE_RSCUNIT	リソースユニット単位
PJCMD_SCOPE_RSCGRP	リソースグループ単位
PJCMD_SCOPE_NODE	ノード単位

失敗した場合は、-1を返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

資源情報*rscinfo_p*が不正(NULL)です。

C.6.16 pjcmd_rscstat_get_rscinfo_scope_value()

```
pjcmd_result_t pjcmd_rscstat_get_rscinfo_scope_value(const PjcmdRscstatRscinfo_t *rscinfo_p, pjcmd_scope_t scope, void *val_p)
```

資源情報について、その取得単位の名称(クラスタ名、リソースユニット名、またはリソースグループ名)またはID(ノードグループID、ブートグループID、またはノードID)を取得します。

[引数]

rscinfo_p

資源情報へのポインター。関数pjcmd_rscstat_get_rscinfo()が返す値です。

scope

情報取得単位のタイプ(下記の表を参照)。
*scope*が資源情報の取得単位のタイプと異なる場合はエラーになります。

val_p

*scope*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を呼び出す側が用意する必要があります。

scope	**val_p*	**val_p*の型
PJCMD_SCOPE_CLUSTER	クラスタ名 **val_p*が指す領域は、応答情報解放後は不定になります。	char *
PJCMD_SCOPE_NODEGRP	ノードグループID	uint32_t
PJCMD_SCOPE_BOOTGRP	ブートグループID	uint32_t
PJCMD_SCOPE_RSCUNIT	リソースユニット名 **val_p*が指す領域は、応答情報解放後は不定になります。	char *
PJCMD_SCOPE_RSCGRP	リソースグループ名 **val_p*が指す領域は、応答情報解放後は不定になります。	char *
PJCMD_SCOPE_NODE	ノードID	uint32_t

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*rscinfo_p*または*val_p*の値が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

*scope*は、この資源情報の取得単位と一致しません。

C.6.17 pjcmd_rscstat_get_rscinfo_info()

```
pjcmd_result_t pjcmd_rscstat_get_rscinfo_info(const PjcmdRscstatRscinfo_t *rscinfo_p, pjcmd_rscstat_rsc_name_t rscname,
pjcmd_rscstat_rsc_value_t type, void *val_p)
```

資源情報から、特定の資源の総量、使用量、または空き量を参照します。

[引数]

rscinfo_p

資源情報へのポインター

rscname

参照する資源名の識別子(下記の表を参照)

type

参照する資源量の識別子(下記の表を参照)

val_p

*rscname*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を呼び出す側が用意する必要があります。

rscname	**val_p*	**val_p*の型
PJCMD_RSCSTAT_RSC_NODE	計算ノード数	uint32_t
PJCMD_RSCSTAT_RSC_CPU	計算ノードのCPUコア数	uint32_t
PJCMD_RSCSTAT_RSC_MEM	計算ノードのメモリ量	uint64_t

type	説明
PJCMD_RSCSTAT_RSC_TOTAL	資源*rscname*の総量
PJCMD_RSCSTAT_RSC_ALLOC	資源*rscname*の使用量
PJCMD_RSCSTAT_RSC_FREE	資源*rscname*の未使用の量

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*rscinfo_p*または*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*rscname*または*type*に不明な値が指定されました。

C.6.18 pjcmd_rscstat_read_jobinfo()

```
PjcmdSubjobid_t *pjcmd_rscstat_read_jobinfo(PjcmdRscstatRscinfo_t *rscinfo_p, pjcmd_rscstat_jobtype_t type)
```

資源情報がノード単位の情報の場合に、そのノード資源を使用して実行されているジョブ、またはそのノードを通信経路として利用しているジョブのサブジョブID構造体を1つ返します。この関数を呼び出すたびに、次の該当するジョブについて返します。

[引数]

rscinfo_p

資源情報へのポインター

type

参照するジョブの種類

識別子	意味
PJCMD_RSCSTAT_RUNNING_JOBS	ノードの資源を使用して実行中のジョブ
PJCMD_RSCSTAT_JOBS_USING_ROUTE	ノードを通信経路として使用しているジョブ

[返り値]

サブジョブID構造体へのポインター。
得られたポインターが指す領域の内容は、次にこの関数を呼び出したあとは不定になります。
失敗した場合はNULLを返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*rscinfo_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*type*に不明が値が指定されました。

PJCMD_ERROR_NODATA

次のジョブはありません。

C.6.19 pjcmd_rscstat_get_rscinfo_customrsc_num()

```
pjcmd_result_t pjcmd_rscstat_get_rscinfo_customrsc_num(const PjcmdRscstatRscinfo_t *rscinfo_p,
pjcmd_rscstat_customrsc_alloc_t type, uint32_t *num_p)
```

資源情報に含まれる、カスタム資源数を取得します。

[引数]

rscinfo_p

資源情報へのポインター

type

カスタム資源の割当タイプ

識別子	意味
PJCMD_RSCSTAT_CUSTOMRSC_ALLOC_RU_RG	リソースユニットまたはリソースグループごとに割り当てるカスタム資源
PJCMD_RSCSTAT_CUSTOMRSC_ALLOC_NODE	ノードごとに割り当てるカスタム資源

num_p

**num_p*にカスタム資源数が格納されます。領域は呼び出し側が確保する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*rscinfo_p*または*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*type*に不明な値が指定されました。

C.6.20 pjcmd_rscstat_get_rscinfo_customrscinfo()

```
PjcmdRscstatCustomRscinfo_t *pjcmd_rscstat_get_rscinfo_customrscinfo(const PjcmdRscstatRscinfo_t *rscinfo_p,
pjcmd_rscstat_customrsc_alloc_t type, uint32_t indx)
```

資源情報に含まれる、特定のカスタム資源の情報を取得します。

[引数]

rscinfo_p

資源情報へのポインター

type

カスタム資源の割当てタイプ

indx

カスタム資源のインデックス。値は0から"関数pjcmd_rscstat_get_rscinfo_customrsc_num()で取得した値-1"の範囲が指定できます。

[返り値]

カスタム資源情報。
失敗時は、NULLを返し、原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*rscinfo_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*type*に不明な値が指定されました。

PJCMD_ERROR_NODATA

*indx*の値が範囲外です。

C.6.21 pjcmd_rscstat_get_customrscinfo_value()

```
pjcmd_result_t pjcmd_rscstat_get_customrscinfo_value(const PjcmdRscstatCustomRscinfo_t *rscinfo_p,
pjcmd_rscstat_customrsc_value_t type, void *val_p)
```

カスタム資源情報の各種情報を参照します。

[引数]

rscinfo_p

カスタム資源情報へのポインター。関数pjcmd_rscstat_get_rscinfo_customrscinfo()で取得した情報です。

type

参照する情報の識別子(下記の表を参照)

val_p

*type*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を呼び出す側が用意する必要があります。

type	**val_p*	**val_p*の型
PJCMD_RSCSTAT_CUSTOMRSC_NAME	カスタム資源の名称 取得した値を応答情報の解放後に参照すると、動作は不定です。	char *
PJCMD_RSCSTAT_CUSTOMRSC_TYPE	カスタム資源の資源量のタイプ ・ PJCMD_RSCSTAT_CUSTOMRSC_NUM カスタム資源は数値型です。 ・ PJCMD_RSCSTAT_CUSTOMRSC_KIND カスタム資源は種別型です。	int
PJCMD_RSCSTAT_CUSTOMRSC_TOTAL	カスタム資源の総量 (カスタム資源が数値型の場合)	int64_t

type	**val_p*	**val_p*の型
PJCMD_RSCSTAT_CUSTOMRSC_ALLOC	使用中のカスタム資源量 (カスタム資源が数値型の場合)	int64_t
PJCMD_RSCSTAT_CUSTOMRSC_FREE	未使用のカスタム資源量 (カスタム資源が数値型の場合)	int64_t
PJCMD_RSCSTAT_CUSTOMRSC_KIND_NUM	カスタム資源の数 (カスタム資源が種別型の場合)	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

*rscinfo_p*または*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*type*に不明な値が指定されました。

C.6.22 pjcmd_rscstat_get_customrscinfo_kind_value()

```
pjcmd_result_t pjcmd_rscstat_get_customrscinfo_kind_value(const PjcmdRscstatCustomRscinfo_t *rscinfo_p, int indx,
pjcmd_rscstat_customrsc_kind_value_t type, void *val_p)
```

カスタム資源情報のタイプが種別型の場合に、その資源情報を参照します。

[引数]

info_p

カスタム資源情報(種別型)へのポインター

indx

参照するカスタム資源情報(種別型)のインデックス。値は、0から"関数pjcmd_rscstat_get_customrscinfo_value()で取得した値-1"の範囲が指定できます。

type

参照する情報の識別子(下記の表を参照)

val_p

*type*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を呼び出す側が用意する必要があります。

type	**val_p*	**val_p*の型
PJCMD_RSCSTAT_CUSTOMRSC_KIND_NAME	カスタム資源の種別の名称	char *
PJCMD_RSCSTAT_CUSTOMRSC_KIND_TOTAL	カスタム資源の総量	int64_t
PJCMD_RSCSTAT_CUSTOMRSC_KIND_ALLOC	使用中のカスタム資源量	int64_t
PJCMD_RSCSTAT_CUSTOMRSC_KIND_FREE	未使用のカスタム資源量	int64_t

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_ARGUMENT

- *rscinfo_p*または*val_p*が不正(NULL)です。
- *rscinfo_p*は、種別型のカスタム資源ではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*type*に不明な値が指定されました。

PJCMD_ERROR_NODATA

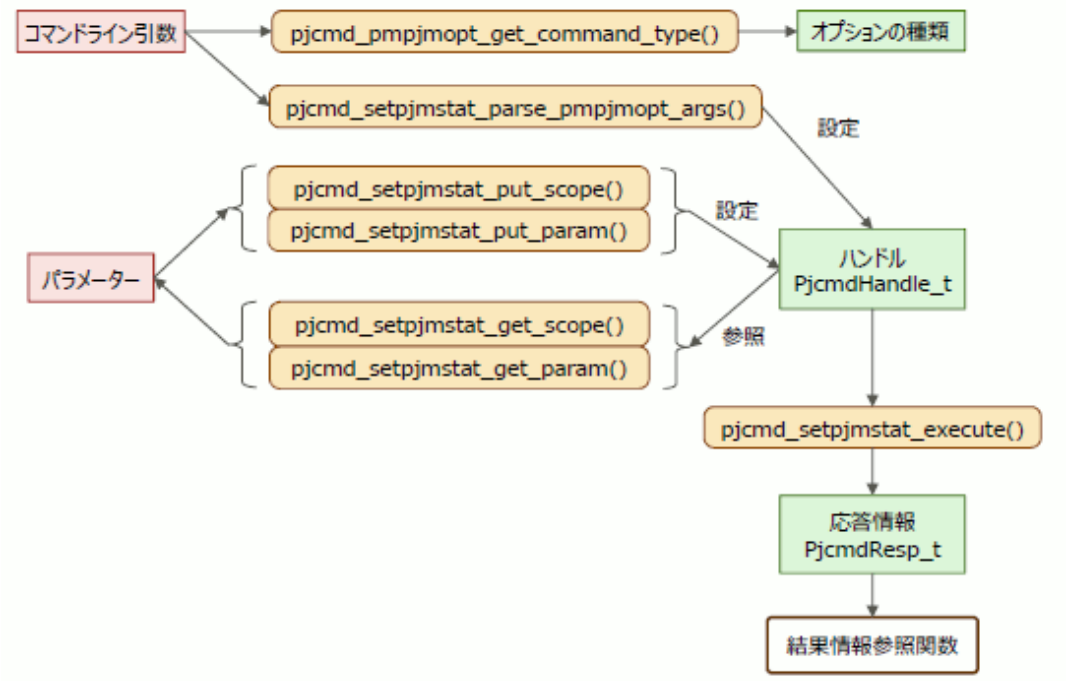
*indx*の値が範囲外です。

付録D　リファレンス：ジョブ運用の設定操作API

D.1　ジョブ投入・実行の許可設定

ここでは、ジョブ運用におけるジョブの投入や実行の許可を設定するための関数を説明します。

図D.1 ジョブの投入・実行の許可の設定依頼



D.1.1　pjcmd_pmpjmopt_get_command_type()

```
pjcmd_pmpjmopt_command_type_t pjcmd_pmpjmopt_get_command_type(int argc, char **argv_pp)
```

コマンドライン引数をpmpjmoptコマンドの引数として解析し、--set-rsc-ugオプションの--show-rsc-ugのどちらが指定されているかを判断します。

[引数]

argc

引数の個数

argv_pp

引数の配列

[返り値]

pmpjmoptコマンドの動作種別。

PJCMD_PMPJMOPT_SET_RSC_UG

--set-rsc-ugオプションが指定されています。

PJCMD_PMPJMOPT_SHOW_RSC_UG

--show-rsc-ugオプションが指定されています。

PCMD_PMPJMOPT_UNKNOWN_COMMAND_TYPE

判断ができません(両オプションが指定されています。またはどちらも指定されていません)。

[pjcmd_errcode]

PJCMD_SUCCESS

成功しました。返り値がPJCMD_PMPJMOPT_SET_RSC_UGおよびPJCMD_PMPJMOPT_SHOW_RSC_UGの場合に設定されます。

PJCMD_ERROR_UNKNOWN_OPTION

判断ができませんでした。返り値がPCMD_PMPJMOPT_UNKNOWN_COMMAND_TYPEの場合に設定されます。

D.1.2 pjcmd_setpjmstat_parse_pmpjmopt_args()

```
pjcmd_result_t pjcmd_setpjmstat_parse_pmpjmopt_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpmpjmoptコマンドの--set-rsc-ugオプションが指定された場合の仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入・実行許可設定用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。
- 排他のオプションが指定されています。

本関数の呼び出しにより、オプションでない引数が 配列*argv_pp*[]の後ろに移動します。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1] がそのオプションを指します。
なお、本関数はpmpjmoptコマンドの--set-rsc-ugオプションおよびそれと同時に指定できるオプションだけを認識します。

D.1.3 pjcmd_setpjmstat_put_scope()

```
pjcmd_result_t pjcmd_setpjmstat_put_scope(PjcmdHandle_t *handle_p, pjcmd_scope_t scope, void *val_p)
```

ジョブ投入・実行を許可する対象範囲をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

scope

対象範囲を示す識別子(下記の表を参照)

val_p

対象範囲を示す値が格納された領域へのポインター。例えば、設定する値の型がchar *型の場合、char *型の値を格納した領域を呼び出し側で用意し、それへのポインター(char **)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

scope	**val_p*	**val_p*の型
PJCMD_SCOPE_CLUSTER	クラスタ名(1つだけ。pmpjmopt -c相当) 設定をしない場合は、環境変数PXMYCLST の値が適用されます。それも設定されていない場合は、関数pjcmd_setpjmstat_execute()がエラーになります。 このパラメーターは、システム管理ノードで呼び出された場合のみ有効になります。システム管理ノード以外では、設定は無視され、呼び出しノードが属するクラスタが適用されます。	char *
PJCMD_SCOPE_RSCUNIT	リソースユニット名(1つだけ。pmpjmopt --rscunit相当) リソースユニット名の指定は必須です。リソースユニット名を指定しない場合は、関数pjcmd_setpjmstat_execute()がエラーになります。	char *
PJCMD_SCOPE_RSCGRP	リソースグループ名の配列(pmpjmopt --rscgrp相当) 最後の要素は(char *)NULLでなければいけません。リソースグループ名"*"は、すべてのリソースグループを意味します(pmpjmopt --all-rsc-groups相当)。 リソースグループを指定しない場合、各リソースグループの設定は、リソースユニットの設定に従います。	char **

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入・実行許可設定用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値または指定できない値が指定されました。

PJCMD_ERROR_INVALID_PARAM

*val_p*が不正(NULL)です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

D.1.4 pjcmd_setpjmstat_get_scope()

```
pjcmd_result_t pjcmd_setpjmstat_get_scope(const PjcmdHandle_t *handle_p, pjcmd_scope_t scope, void *val_p)
```

ハンドルに設定されているジョブ投入・実行の許可の対象範囲(クラスタ、リソースユニット、リソースグループ)を参照します。

[引数]

handle_p

ハンドルへのポインター

scope

参照する対象範囲の識別子。指定できる識別子は、関数pjcmd_setpjmstat_put_scope()と同じです。

val_p

*scope*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入・実行許可設定用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値または指定できない値が指定されました。

PJCMD_ERROR_NODATA

指定した*scope*は、ハンドルに設定されていません。

D.1.5 pjcmd_setpjmstat_put_param()

```
pjcmd_result_t pjcmd_setpjmstat_put_param(PjcmdHandle_t *handle_p, pjcmd_setpjmstat_param_t param, const void *val_p)
```

ジョブ投入・実行の許可に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

ジョブ投入・実行の許可に関するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。例えば、設定する値の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_SETPJMSTAT_JOB_SUBMIT	ジョブ投入の許可。 0: ジョブの新規投入を許可しません。 1: ジョブの新規投入を許可します。 未設定時はジョブ投入の許可は変更しません。	int
PJCMD_SETPJMSTAT_JOB_EXECUTE	ジョブ実行の許可。 0: ジョブの新規実行を許可しません。 1: ジョブの新規実行を許可します。 未設定時はジョブ実行の許可は変更しません。	int
PJCMD_SETPJMSTAT_HELP	pmpjmoptコマンドの--helpオプションの指定に相当。 0 : 指定なし(デフォルト)。 1 : 指定あり。 このパラメーターは、ジョブ投入や実行の許可には影響しません。	int

パラメーターPJCMD_SETPJMSTAT_JOB_SUBMITおよびPJCMD_SETPJMSTAT_JOB_EXECUTEのどちらも設定されていない場合は、関数pjcmd_setpjmstat_execute()呼び出し時にエラーになります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入・実行許可設定用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法の間違い
- 間違った値

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

D.1.6 pjcmd_setpjmstat_get_param()

```
pjcmd_result_t pjcmd_setpjmstat_get_param(const PjcmdHandle_t *handle_p, pjcmd_setpjmstat_param_t param, void *val_p)
```

ハンドルに設定されているジョブ投入・実行の許可に関するパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_setpjmstat_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入・実行許可設定用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

D.1.7 pjcmd_setpjmstat_execute()

```
PjcmdResp_t *pjcmd_setpjmstat_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、ジョブ投入・実行の許可をジョブ運用管理機能に依頼します。この関数の呼び出しにはroot権限が必要です。

[引数]

handle_p

ハンドルへのポインター

[返り値]

ジョブ投入・実行の許可の応答情報。
得られた応答情報は、pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。ジョブ投入・実行の許可の依頼が失敗した場合は、NULLを返し、pjcmd_errcodeが設定されます。
なお、応答情報は依頼が成功したか否かを示します。ジョブ投入・実行の許可が正常に受け付けられたかどうかは、応答情報に基づいて関数pjcmd_get_result()で結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。

- ジョブ投入・実行許可設定用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。この関数は、システム管理ノードだけで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

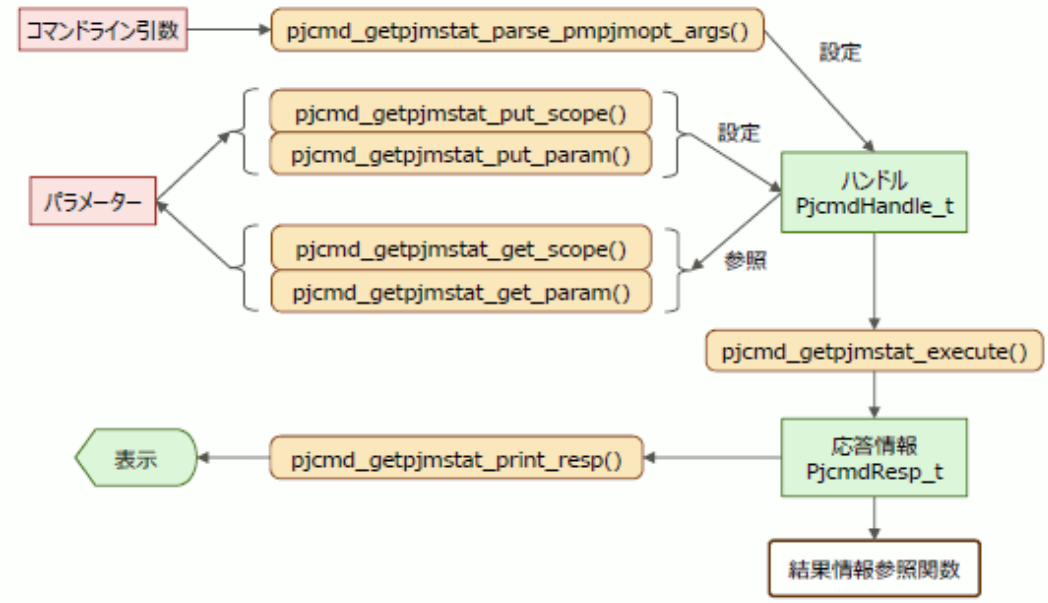
PJCMD_ERROR_INTERNAL

内部エラー

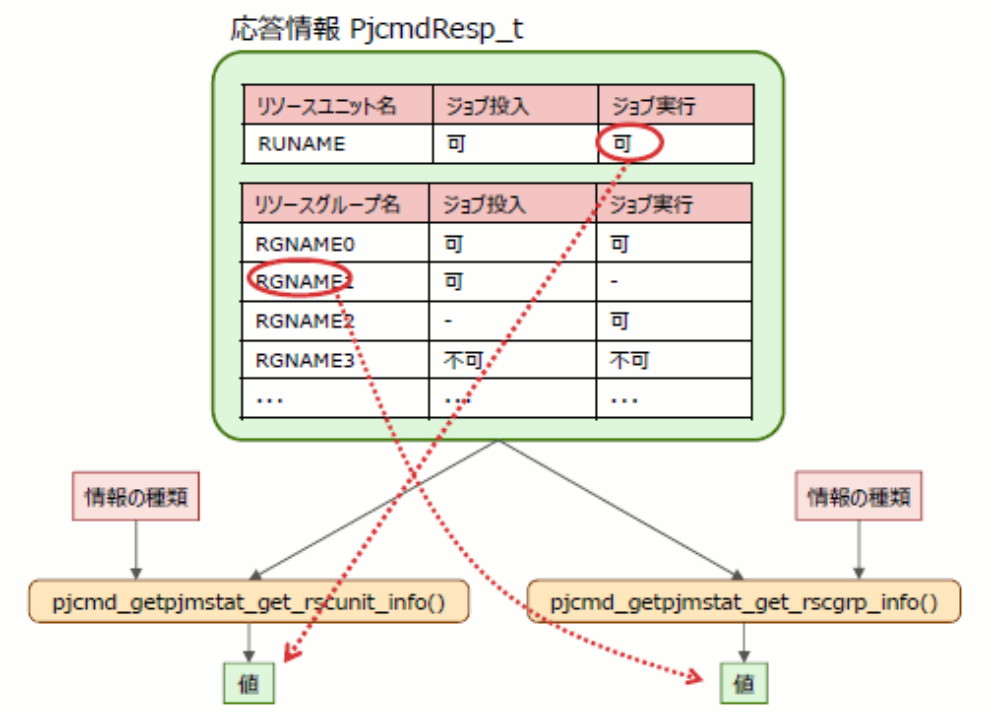
D.2 ジョブ投入・実行の許可情報の参照

ここでは、ジョブ運用におけるジョブの投入や実行の許可情報を参照するための関数を説明します。

図D.2 ジョブの投入・実行の許可情報の取得依頼



図D.3 ジョブの投入・実行の許可情報の参照



D.2.1 pjcmd_getpjmstat_parse_pmpjmopt_args()

```
pjcmd_result_t pjcmd_getpjmstat_parse_pmpjmopt_args(PjcmdHandle_t *handle_p, int argc, char **argv_pp)
```

コマンドライン引数をpmpjmoptコマンドの--show-rsc-ugオプションが指定された場合の仕様に従って解析し、指定内容をハンドルへ設定します。

[引数]

handle_p

ハンドルへのポインター

argc

引数の数

argv_pp

引数の配列

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入・実行許可情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*argc*または*argv_pp*が正しくありません。

PJCMD_ERROR_UNKNOWN_OPTION

不明なオプションを検出しました。

PJCMD_ERROR_INVALID_OPTION

オプションの指定方法が正しくありません。

- オプションの引数の指定方法が間違っています。
- オプションに必須の引数が指定されていません。
- 排他のオプションが指定されています。

本関数の呼び出しにより、オプションでない引数が 配列*argv_pp*[]の後ろに移動します。
認識できないオプションを検出した場合は、引数の解析を終了し、*argv_pp*[pjcmd_optind-1] がそのオプションを指します。
なお、本関数はpmpjmoptコマンドの--show-rsc-ugオプションおよびそれと同時に指定できるオプションだけを認識します。

D.2.2 pjcmd_getpjmstat_put_scope()

```
pjcmd_result_t pjcmd_getpjmstat_put_scope(PjcmdHandle_t *handle_p, pjcmd_scope_t scope, const void *val_p)
```

ジョブ投入・実行の許可情報を取得する対象範囲をハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

scope

対象範囲を示す識別子(下記の表を参照)

val_p

対象範囲を示す値が格納された領域へのポインター。例えば、設定する値の型がchar *型の場合、char *型の値を格納した領域を呼び出し側で用意し、それへのポインター(char **)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

scope	**val_p*	**val_p*の型
PJCMD_SCOPE_CLUSTER	クラスタ名(1つだけ。pmpjmopt -c相当) 設定をしない場合は、環境変数PXMYCLST の値が適用されます。それも設定されていない場合は、関数pjcmd_getpjmstat_execute()がエラーになります。 このパラメーターは、システム管理ノードで呼び出された場合のみ有効になります。システム管理ノード以外では、設定は無視され、呼び出しノードが属するクラスタが適用されます。	char *
PJCMD_SCOPE_RSCUNIT	リソースユニット名(1つだけ。pmpjmopt --rscunit相当) リソースユニット名の設定は必須です。リソースユニット名を設定しない場合は、関数pjcmd_getpjmstat_execute()がエラーになります。	char *
PJCMD_SCOPE_RSCGRP	リソースグループ名の配列(pmpjmopt --rscgrp相当) 最後の要素は(char *)NULLでなければいけません。リソースグループ名"*"は、すべてのリソースグループを意味します。 リソースグループの設定をしない場合、リソースユニットごとの情報を取得します。	char **

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入・実行許可情報取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値または指定できない値が指定されました。

PJCMD_ERROR_INVALID_PARAM

*val_p*が不正(NULL)です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

D.2.3 pjcmd_getpjmstat_get_scope()

```
pjcmd_result_t pjcmd_getpjmstat_get_scope(const PjcmdHandle_t *handle_p, pjcmd_scope_t scope, void *val_p)
```

ハンドルに設定されているジョブ投入・実行の許可情報の取得対象(クラスタ、リソースユニット、リソースグループ)を参照します。

[引数]

handle_p

ハンドルへのポインター

scope

参照する対象範囲の識別子。指定できる識別子は、関数pjcmd_getpjmstat_put_scope()と同じです。

val_p

*scope*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入・実行許可情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*scope*に不明な値または指定できない値が指定されました。

PJCMD_ERROR_NODATA

指定した*scope*は、ハンドルに設定されていません。

D.2.4 pjcmd_getpjmstat_put_param()

```
pjcmd_result_t pjcmd_getpjmstat_put_param(PjcmdHandle_t *handle_p, pjcmd_getpjmstat_param_t param, const void *val_p)
```

ジョブ投入・実行の許可情報取得に関するパラメーターをハンドルに設定します。

[引数]

handle_p

ハンドルへのポインター

param

ジョブ投入・実行の許可情報取得に関するパラメーターの識別子(下記の表を参照)

val_p

設定するパラメーターの値が格納された領域へのポインター。設定する値の型がint型の場合、int型の値を格納した領域を呼び出し側で用意し、それへのポインター(int *)を*val_p*に指定してください。NULLを指定した場合、パラメーターの値を初期状態(未設定)に戻します。

param	**val_p*	**val_p*の型
PJCMD_GETPJMSTAT_HELP	pmpjmoptコマンドの--helpオプションの指定に相当。 0 : 指定なし(デフォルト)。 1 : 指定あり。 このパラメーターは、ジョブ投入・実行の許可情報取得には影響しません。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入・実行許可情報取得用のハンドルではありません。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

パラメーターの値が不正です。

- 指定方法の間違い
- 間違った値

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

D.2.5 pjcmd_getpjmstat_get_param()

```
pjcmd_result_t pjcmd_getpjmstat_get_param(const PjcmdHandle_t *handle_p, pjcmd_getpjmstat_param_t param, void *val_p)
```

ハンドルに設定されているジョブ投入・実行の許可情報取得に関するパラメーターを参照します。

[引数]

handle_p

ハンドルへのポインター

param

参照するパラメーターの識別子。指定できる識別子は、関数pjcmd_getpjmstat_put_param()と同じです。

val_p

*param*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入・実行許可情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*param*に不明な値が指定されました。

PJCMD_ERROR_NODATA

指定したパラメーターは、ハンドルに設定されていません。

D.2.6 pjcmd_getpjmstat_execute()

```
PjcmdResp_t *pjcmd_getpjmstat_execute(const PjcmdHandle_t *handle_p)
```

ハンドルに基づいて、ジョブ投入・実行の許可状態の取得をジョブ運用管理機能へ依頼します。この関数の呼び出しにはroot権限が必要です。

[引数]

handle_p

ハンドルへのポインター

[返り値]

ジョブ投入・実行の許可情報取得の応答情報。
得られた応答情報は、pjcmd_destroy_resp()を利用して、呼び出し側が解放しなければいけません。情報取得の依頼に失敗した場合は、NULLを返し、pjcmd_errcodeが設定されます。

なお、応答情報は依頼が成功したか否かを示します。依頼が正常に受け付けられたかどうかは、応答情報に基づいて関数pjcmd_get_result()で結果コードを調べる必要があります。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_HANDLE

ハンドルが不正です。

- *handle_p*がNULLです。
- ジョブ投入・実行許可情報取得用のハンドルではありません。

PJCMD_ERROR_INVALID_NODE

このノードでは、この関数は呼び出せません。この関数は、システム管理ノードだけで呼び出せます。

PJCMD_ERROR_INVALID_PARAM

ハンドル内のパラメーターが不正です。

PJCMD_ERROR_NOMEM

メモリの獲得に失敗しました。

PJCMD_ERROR_NOPERM

関数の呼び出しが許可されていません。

PJCMD_ERROR_INTERNAL

内部エラー

D.2.7 pjcmd_getpjmstat_print_resp()

```
pjcmd_result_t pjcmd_getpjmstat_print_resp(const PjcmdResp_t *resp_p)
```

ジョブ投入・実行の許可情報取得の結果を、pmpjmoptコマンドの仕様に従って標準出力に出力します。

[引数]

resp_p

応答情報へのポインター

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- ジョブ投入・実行許可情報取得の応答情報ではありません。
- 情報の取得が成功した応答情報ではありません。

D.2.8 pjcmd_getpjmstat_get_rscunit_info()

```
pjcmd_result_t pjcmd_getpjmstat_get_rscunit_info(const PjcmdResp_t *resp_p, pjcmd_getpjmstat_info_t type, void *val_p)
```

ジョブ投入・実行の許可情報取得の応答情報から、リソースユニットについての情報を参照します。

[引数]

resp_p

応答情報へのポインター

type

参照する情報の種類の識別子(下記の表を参照)

val_p

*type*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

type	**val_p*	**val_p*の型
PJCMD_GETPJMSTAT_INFO_RSCUNIT	リソースユニット名 **val_p*が指す領域は、応答情報解放後は不定になります。	char *
PJCMD_GETPJMSTAT_INFO_JOB_SUBMIT	ジョブの投入許可 0: ジョブの新規投入は許可されていません。 1: ジョブの新規投入は許可されています。 -1: 不明	int
PJCMD_GETPJMSTAT_INFO_JOB_EXECUTE	ジョブの実行許可 0: ジョブの新規実行は許可されていません。 1: ジョブの新規実行は許可されています。 -1: 不明	int
PJCMD_GETPJMSTAT_INFO_RSCGRP_NUM	リソースユニット内のリソースグループ数 情報の取得対象がリソースユニットの場合は、0になります。	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- ジョブ投入・実行許可情報取得の応答情報ではありません。
- 応答情報はリソースユニットについての結果ではなく、リソースグループについての結果です。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*type*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

*type*の値はこの関数では指定できません。

D.2.9 pjcmd_getpjmstat_get_rscgrp_info()

```
pjcmd_result_t pjcmd_getpjmstat_get_rscgrp_info(const PjcmdResp_t *resp_p, int indx, pjcmd_getpjmstat_info_t info, void *val_p)
```

ジョブ投入・実行の許可情報取得の応答情報から、特定のリソースグループの情報を参照します。

[引数]

resp_p

応答情報へのポインター

indx

取得するリソースグループのインデックス。
値は、0から"関数pjcmd_getpjmstat_get_rscunit_info()で取得するリソースグループ数-1"の範囲が指定できます。

info

参照する情報の識別子(下記の表を参照)

val_p

*info*に応じた型で、**val_p*に値が格納されます。値の型に応じたサイズの領域を、呼び出す側が用意する必要があります。

info	**val_p*	**val_p*の型
PJCMD_GETPJMSTAT_INFO_RSCGRP	リソースグループ名	char *
PJCMD_GETPJMSTAT_INFO_JOB_SUBMIT	ジョブの投入許可 0: ジョブの新規投入は許可されていません。 1: ジョブの新規投入は許可されています。 -1: 不明	int
PJCMD_GETPJMSTAT_INFO_JOB_EXECUTE	ジョブの実行許可 0: ジョブの新規実行は許可されていません。 1: ジョブの新規実行は許可されています。 -1: 不明	int

[返り値]

PJCMD_OK

成功

PJCMD_ERR

失敗。原因がpjcmd_errcodeに設定されます。

[pjcmd_errcode]

PJCMD_ERROR_INVALID_RESP

応答情報が不正です。

- *resp_p*がNULLです。
- ジョブ投入、実行可否取得の応答情報ではありません。
- リソースグループの状態取得結果ではなく、リソースユニットの状態取得結果です。

PJCMD_ERROR_INVALID_ARGUMENT

*val_p*が不正(NULL)です。

PJCMD_ERROR_UNKNOWN_PARAM

*info*に不明な値が指定されました。

PJCMD_ERROR_INVALID_PARAM

*info*の値はこの関数では指定できません。

PJCMD_ERROR_NODATA

*indx*の値が範囲外です。

付録E　サンプルプログラム

ここでは、コマンドAPIを呼び出すサンプルプログラムを示します。

サンプルプログラムのソースファイルはログインノード、計算クラスタ管理ノード、およびシステム管理ノードの以下のディレクトリにインストールされています。

```
/usr/src/FJSVtcs/pjm/pjcmd/
```

サンプルプログラムのコンパイル方法については、上記ディレクトリ配下にあるドキュメントを参照してください。

E.1 ジョブの投入

/usr/src/FJSVtcs/pjm/pjcmd/c/submit/pjcmd_submit.c

```
/*
 * ジョブ投入
 */

#include <stdio.h>
#include <stdlib.h>
#include <FJSVtcs/pjm/pjcmd.h>

#define EXIT_SUCCESS 0
#define EXIT_FAILURE 1

const char *CMD_NAME = "pjsub_custom";

int main(int argc, char **argv)
{
    PjcmdHandle_t *handle_p;
    PjcmdResp_t *resp_p;
    char *script_p;
    int i, code;
    int32_t line;
    char *detail_p;
    PjcmdSubjobid_t *subjobid_p;
    char subjobid_str[PJCMD_MAX_SUBJOBID_STR_LEN];

    /* ジョブ投入ハンドルを作成 */
    handle_p = pjcmd_create_handle(PJCMD_SUBMIT);
    if (handle_p == NULL) {
        fprintf(stderr, "%s: Failed in create_handle : %s¥n", CMD_NAME, pjcmd_strerror(pjcmd_errcode));
        exit(EXIT_FAILURE);
    }

    /* コマンドライン引数を解析し、ハンドルに設定 */
    if (pjcmd_submit_parse_pjsub_args(handle_p, argc, argv) == PJCMD_ERR) {
        /* 解析に失敗した引数を表示して終了 */
        fprintf(stderr, "%s: Failed in parse_args : %s¥n", CMD_NAME, argv[pjcmd_optind - 1]);
        exit(EXIT_FAILURE);
    }

    /* ジョブスクリプト内の指示行を解析し、ハンドルに設定
     * pjcmd_optind は残りの引数＝スクリプトを指している。
     * (簡単のために、スクリプトは1つだけとする)
     */
    if (pjcmd_optind == argc) {
        fprintf(stderr, "%s: job script does not specified¥n", CMD_NAME);
        pjcmd_destroy_handle(handle_p);
        exit(EXIT_FAILURE);
    }
```

```
    script_p = argv[pjcmd_optind];
    if (pjcmd_submit_parse_pjsub_scriptfile(handle_p, script_p, "#PJM", &line, &detail_p) == PJCMD_ERR) {
        fprintf(stderr, "%s: Failed to parse script %s (line=%d, arg=%s)¥n", CMD_NAME, script_p, line, detail_p);
        pjcmd_destroy_handle(handle_p);
        exit(EXIT_FAILURE);
    }

    /* スクリプトファイル名をハンドルに設定 */
    if (pjcmd_submit_put_param(handle_p, PJCMD_SUBMIT_SCRIPTFILE, &script_p) == PJCMD_ERR) {
        fprintf(stderr, "%s: Failed in setting script name : %s¥n", CMD_NAME, script_p);
        pjcmd_destroy_handle(handle_p);
        exit(EXIT_FAILURE);
    }

    /* ハンドルの情報を使い、ジョブ投入 */
    resp_p = pjcmd_submit_execute(handle_p);
    if (resp_p == NULL) { /* ジョブ投入に失敗した場合 */
        fprintf(stderr, "%s: Failed in submitting a job : %s¥n", CMD_NAME, script_p);
        pjcmd_destroy_handle(handle_p);
        exit(EXIT_FAILURE);
    }

    /* 投入結果を表示 */
    int64_t jobnum[2];
    pjcmd_get_jobresult_num(resp_p, jobnum);
    for (i = 0; i < jobnum[0]; i++) {
        /* サブジョブID構造体を取得し、文字列に変換 */
        pjcmd_get_jobresult_info(resp_p, PJCMD_JOBRESULT_ANY, i, PJCMD_JOBRESULT_SUBJOBID, &subjobid_p);
        pjcmd_subjobid_to_str(subjobid_p, subjobid_str);
        /* 結果コード取得 */
        pjcmd_get_jobresult_info(resp_p, PJCMD_JOBRESULT_ANY, i, PJCMD_JOBRESULT_CODE, &code);
        /* 結果表示 */
        printf("Job %s : %s¥n", subjobid_str, (code == 0) ? "submitted" : "submit failed");
    }
    /* 応答情報の解放 */
    pjcmd_destroy_resp(resp_p);
    /* ハンドルの解放 */
    pjcmd_destroy_handle(handle_p);
    exit(EXIT_SUCCESS);
}
```

E.2 ジョブ情報の取得

/usr/src/FJSVtcs/pjm/pjcmd/c/jobinfo/pjcmd_jobinfo.c

```
/*
 * ジョブの状態表示
 */

#include <stdio.h>
#include <stdlib.h>
#include <FJSVtcs/pjm/pjcmd.h>

#define EXIT_SUCCESS 0
#define EXIT_FAILURE 1

const char *CMD_NAME = "pjstat_custom";

int main(void)
{
    PjcmdHandle_t *handle_p;
```

```
PjcmdResp_t *resp_p;

/* ジョブ情報取得用のハンドルを作成 */
handle_p = pjcmd_create_handle(PJCMD_JOBINFO);
if (handle_p == NULL) {
    fprintf(stderr, "%s: Failed to create handle : %s¥n", CMD_NAME, pjcmd_strerror(pjcmd_errcode));
    exit(EXIT_FAILURE);
}

/* リソースユニット・リソースグループ単位で集計した情報を取得 */
const char *grouping_items_p[2] = { "rscu", "rscg" };
pjcmd_jobinfo_put_condition(handle_p, PJCMD_JOBINFO_GROUPING, grouping_items_p, 2);

/* 取得するジョブの条件を以下のように設定
 *  - 状態はRUN (pjstat --filter st=RUN)
 *  - ジョブ名はfooで始まる (pjstat --filter "jname=foo*")
 *  - ジョブIDは10以下 (pjstat --filter jid=-10)
 *  - 優先度は5以上 (pjstat --filter prio=5-)
 *  - elapse limit は1時間以上2時間以下
 *    (pjstat --filter elpl=1:00:00-2:00:00)
 */
const char *filter_exprs_p[5] = { "st=RUN", "jnam=foo*", "jid=-10", "prio=5-", "elpl=1:00:00-2:00:00" };
pjcmd_jobinfo_put_condition(handle_p, PJCMD_JOBINFO_FILTER, filter_exprs_p, 5);

/* 取得項目をジョブID、ジョブ名、ジョブ状態に設定
 * (pjstat --choose jid,jnam,st)
 */
const char *choose_items_p[3] = { "jid", "jnam", "st" };
pjcmd_jobinfo_put_condition(handle_p, PJCMD_JOBINFO_CHOOSE, choose_items_p, 3);

/* 投入時刻昇順にソート */
const char *sort_items_p[] = { "adt:A" };
pjcmd_jobinfo_put_condition(handle_p, PJCMD_JOBINFO_SORT, sort_items_p, 1);

/* ヒストリ（終了済みジョブ）についても
 * 過去5日分については情報を取得する (pjstat -H=day)
 */
int hist = 5;
pjcmd_jobinfo_put_param(handle_p, PJCMD_JOBINFO_HISTORY_DAY, &hist);

/* サマリ情報も取得 (pjstat --with-summary 相当) */
int summary = PJCMD_JOBINFO_WITH_SUMMARY;
pjcmd_jobinfo_put_param(handle_p, PJCMD_JOBINFO_SUMMARY, &summary);

/* サブジョブ情報も取得 (pjstat -E 相当) */
int verbose = PJCMD_JOBINFO_VERBOSITY_SUBJOB;
pjcmd_jobinfo_put_param(handle_p, PJCMD_JOBINFO_VERBOSITY, &verbose);

/*  他ユーザーのジョブ情報も取得する(pjstat -A相当)
 *  (一般ユーザーの権限でコマンド呼び出しされた場合、
 *   ACL設定により参照できない項目はマスクされる)
 */
int othersjob = PJCMD_JOBINFO_OTHERSJOB_ALL;
pjcmd_jobinfo_put_param(handle_p, PJCMD_JOBINFO_OTHERSJOB, &othersjob);

/* ハンドルの内容に基づいて、ジョブ情報の取得を依頼 */
resp_p = pjcmd_jobinfo_execute(handle_p);
if (resp_p == NULL) {
    fprintf(stderr, "%s: Request failed¥n", CMD_NAME);
    pjcmd_destroy_handle(handle_p);
    exit(EXIT_FAILURE);
}
int code;
```

```
    int subcode;
    char *detail_p;
    pjcmd_get_result(resp_p, &code, &subcode, &detail_p);
    if (code != 0) {
        fprintf(stderr, "%s: Request failed (code=%d)¥n", CMD_NAME, code);
        pjcmd_destroy_handle(handle_p);
        exit(EXIT_FAILURE);
    }

    /* 応答情報 resp_p から、情報グループを１つずつ読む
     *（取得条件より、ジョブ情報グループはリソースユニット・リソースグループ単位）
     */
    pjcmd_result_t ret;
    while (1) {
        ret = pjcmd_jobinfo_read_infogrp(resp_p);
        if (ret == PJCMD_ERR) {
            if (pjcmd_errcode == PJCMD_ERROR_NODATA) {
                break; /* すべての情報グループを読んだ */
            }
            fprintf(stderr, "%s: Cannot read infogrp¥n", CMD_NAME);
            pjcmd_destroy_resp(resp_p);
            pjcmd_destroy_handle(handle_p);
            exit(EXIT_FAILURE);
        }

        /* 情報グループのサマリ行を表示 */
        pjcmd_jobinfo_print_resp(resp_p, PJCMD_JOBINFO_PRINT_SUMMARY);
        /* 情報グループ内のジョブ情報を表示 */
        pjcmd_jobinfo_print_resp(resp_p, PJCMD_JOBINFO_PRINT_JOBINFO);
    }

    pjcmd_destroy_resp(resp_p);
    pjcmd_destroy_handle(handle_p);
    exit(EXIT_SUCCESS);
}
```

E.3 ジョブの削除

/usr/src/FJSVtcs/pjm/pjcmd/c/kill/pjcmd_kill.c

```
/*
 * ジョブ削除
 */

#include <stdio.h>
#include <stdlib.h>
#include <FJSVtcs/pjm/pjcmd.h>

#define EXIT_SUCCESS 0
#define EXIT_FAILURE 1

const char *CMD_NAME = "pjdel_custom";

int main(int argc, char **argv)
{
    PjcmdHandle_t *handle_p;
    PjcmdResp_t *resp_p;
    PjcmdSubjobid_t *subjobid_p;
    char subjobid_str_p[PJCMD_MAX_SUBJOBID_STR_LEN];
    int i;
    int64_t num[2], total_num, ok_num, err_num, cnt;
    int help_flag;
```

```
/* ジョブ削除ハンドルを作成 */
handle_p = pjcmd_create_handle(PJCMD_KILL);
if (handle_p == NULL) {
    fprintf(stderr, "Failed in create_handle : %s¥n", pjcmd_strerror(pjcmd_errcode));
    exit(EXIT_FAILURE);
}

/* コマンドライン引数を解析 */
if (pjcmd_kill_parse_pjdel_args(handle_p, argc, argv) == PJCMD_ERR) {
    fprintf(stderr, "Failed in parse_args : %s : arg=%s¥n",
                    pjcmd_strerror(pjcmd_errcode), argv[pjcmd_optind - 1]);
    /* pjdel の usage 表示 */
    pjcmd_print_stdcmd_usage(PJCMD_STDCMD_PJDEL, CMD_NAME);
    pjcmd_destroy_handle(handle_p);
    exit(EXIT_FAILURE);
}

/* --help オプション指定時は、usage を表示して終了 */
pjcmd_kill_get_param(handle_p, PJCMD_KILL_HELP, &help_flag);
if (help_flag != 0) {
    /* pjdel の usage 表示 */
    pjcmd_print_stdcmd_usage(PJCMD_STDCMD_PJDEL, CMD_NAME);
    pjcmd_destroy_handle(handle_p);
    exit(EXIT_SUCCESS);
}

/* 引数の残りを削除対象のジョブIDとみなして、ハンドルに設定 */
if (pjcmd_optind == argc) { /* ジョブID なし */
    fprintf(stderr, "Job id is not spesified.¥n");
    pjcmd_destroy_handle(handle_p);
    exit(EXIT_FAILURE);
}
for (i = pjcmd_optind; i < argc; i++) {
    if (pjcmd_put_job_by_str(handle_p, argv[i]) == PJCMD_ERR) {
        fprintf(stderr, "Failed in put_job_by_str : %s¥n", argv[i]);
        pjcmd_destroy_handle(handle_p);
        exit(EXIT_FAILURE);
    }
}

/* ジョブの削除要求 */
resp_p = pjcmd_kill_execute(handle_p);
if (resp_p == NULL) { /* 要求に失敗した場合 */
    fprintf(stderr, "Failed in kill_execute : %s¥n", pjcmd_strerror(pjcmd_errcode));
    pjcmd_destroy_handle(handle_p);
    exit(EXIT_FAILURE);
}

/* 削除結果表示 */
pjcmd_get_jobresult_num(resp_p, num);
total_num = num[0];
ok_num    = num[1];
err_num   = total_num - ok_num;

if (err_num) { /* エラージョブあり */
    fprintf(stderr, "Operation failed for %ld jobs.¥n", err_num);
}
/* 失敗したジョブについてだけ、ジョブID(サブジョブID)を１件１行で表示 */
for (cnt = 0; cnt < err_num; cnt++) {
    pjcmd_get_jobresult_info(resp_p, PJCMD_JOBRESULT_ERR, cnt, PJCMD_JOBRESULT_SUBJOBID, &subjobid_p);
    pjcmd_subjobid_to_str(subjobid_p, subjobid_str_p);
    fprintf(stderr, "Failed for the job %s¥n", subjobid_str_p);
```

```
    }

    pjcmd_destroy_resp(resp_p);
    pjcmd_destroy_handle(handle_p);
    exit((err_num == 0) ? EXIT_SUCCESS : EXIT_FAILURE);
}
```