ITA_利用手順マニュアル

RestAPI

ー第1.3版 ー

免責事項

本書の内容はすべて日本電気株式会社が所有する著作権に保護されています。
本書の内容の一部または全部を無断で転載および複写することは禁止されています。
本書の内容は将来予告なしに変更することがあります。
日本電気株式会社は、本書の技術的もしくは編集上の間違い、欠落について、一切責任を負いません。
日本電気株式会社は、本書の内容に関し、その正確性、有用性、確実性その他いかなる保証もいたしません。

商標

- LinuxはLinus Torvalds氏の米国およびその他の国における登録商標または商標です。
- Red Hatは、Red Hat, Inc.の米国およびその他の国における登録商標または商標です。
- Apache、Apache Tomcat、Tomcatは、Apache Software Foundationの登録商標または商標です。
- Ansibleは、Red Hat, Inc.の登録商標または商標です。
- Active Directoryは、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

その他、本書に記載のシステム名、会社名、製品名は、各社の登録商標もしくは商標です。

なお、® マーク、TMマークは本書に明記しておりません。
※本書では「Exastro IT Automation」を「ITA」として記載します。

目次

はじめに

本書は、ITA システムの RestAPI の概要および操作方法について説明します。

1　ITA システム　REST API の概要

本章では ITA を操作するための標準 REST API について説明します。

1.1　REST API について

ITA では、外部プログラムから ITA で管理されているリソースに対して、各種操作を行うことが可能な REST API を提供しています。

- 以下の表に記載しているメニュー以外はすべて標準的な RESTAPI を使用できます。
  標準的な RESTAPI について、「2 標準 REST 機能の利用」を参照

表 1-1 個別の REST API 一覧

メニューグループ	メニュー名	メニューID	参照先
管理コンソール	メニューエクスポート	2100000211	3 メニューエクスポート/インポート利用編
	メニューインポート	2100000212	
基本コンソール	Symphony 編集	2100000306	4 Symphony 利用編
	Symphony 作業実行	2100000308	
	Symphony 作業確認	2100000309	
Ansible-Legacy	作業実行	2100020111	5 Movement 利用編
	作業状態確認	2100020112	
Ansible-Pioneer	作業実行	2100020211	
	作業状態確認	2100020212	
Ansible-LegacyRole	作業実行	2100020312	
	作業状態確認	2100020313	
DSC	作業実行	2100060009	
	作業状態確認	2100060010	
OpenStack	作業実行	2100070004	
	作業状態確認	2100070005	

2　標準 REST 機能の利用

外部プログラムから REST API を利用して、ITA で管理されているリソースに対し、操作を行うことが可能です。
以下に、呼び出し規約を示します。

2.1　リクエストの形式

ITA の REST API では、ITA 上の各メニューのパスに対して HTTP リクエストを発行します。

パス：
https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=（各メニューのメニューID）
例）[管理コンソール]-[システム設定]メニュー（メニューID：2100000202）の場合
https:// exastro-it-automation:443/default/menu/07_rest_api_ver1.php?no=2100000202
※<HostName>:ITA インストーラでインストールした時のホスト名「exastro-it-automation」

HTTP ヘッダ：
以下の表にあるものが利用可能です。

表 2-1 HTTP ヘッダパラメーター一覧

HTTP ヘッダ	説明
Host	ITA の RestAPI サーバーのホスト名または IP アドレスとポート番号をコロン(:)区切りで指定する。
Method	原則：POST、を指定すること。 例外：ITA で認証不要と設定したメニューにアクセスする場合のみ、GET、を指定することが可能。
Content-Type	“application/json”を指定する。 Method が GET の場合は指定しなくてもよい。
Authorization	ITA の認証要メニューにアクセスする場合は、 「ログイン ID」と「パスワード」* を、半角コロン(:)で結合して、base64encode をした値、を指定。 Method が GET の場合は指定しなくてもよい。
X-Command	Method が POST の場合のみに設定可能。 【INFO】、【FILTER】、【EDIT】 のいずれかを設定できる。

HTTP ヘッダの文字列は大文字でも小文字でも問題ありません。

* ITA のパスワードが期限切れとなっていた場合、RestAPI は Error となります。
Web システムのログイン画面から、パスワードを変更してからリクエストを行ってください。
但し、ActiveDirectory 連携機能を利用している場合は、ActiveDirectory 上で管理される認証情報に従います。（※ActiveDirectory 連携機能の連携対象外ユーザーは、この限りではありません）
ActiveDirectory 連携機能の詳細は、「利用手順マニュアル_管理コンソール」-「ActiveDirectory 連携機能の利用」をご参照ください。

HTTP ヘッダの例：
ログイン ID が[test_loginid]で、パスワードが[test_password]の場合
test_loginid: test_password　を、base64encode で暗号化
→[qTImqS9fo2qcozyxBaEyp3EspTSmp3qipzD=]）

```
Host:<HostName>:<Port>
Content-Type:application/json
Authorization: qTlmqS9fo2qcozyxBaEyp3EspTSmp3qipzD=
X-Command: INFO
```

2.2 利用可能なメソッドとコマンド

利用可能なメソッドとコマンドの階層は以下のとおりです。

```
Method
        ├GET                    -  ①
        │
        └POST
(X-Command) |
              ├【INFO】         -  ②
              ├【FILTER】       -  ③
              └【EDIT】         -  ④
```

(1) GET(Method)

列情報(列番号と列名)および、通常ステータス(廃止または活性中)の全レコードの行数とレコード内容を返却します。

・HTTP ヘッダ

表 2-2 HTTP ヘッダパラメーター一覧

HTTP ヘッダ	値
Method	GET

・content パラメータ

なし

・レスポンス

1)レコード行数

(JSON 形式)

キー{resultdata} -> キー{CONTENTS} -> キー{RECORD_LENGTH}の中に、数値として格納されます。

2)列情報(列番号と列名)

(JSON 形式)

キー{resultdata} -> キー{CONTENTS} -> キー{BODY} -> キー{0}の中に、0 から始まる数値を、キーとする配列として格納されます。

表 2-3 レスポンスパラメーター一覧(列情報)

列番号	列名
0	一列目
1	二列目
⋮	⋮

3)レコード情報

(JSON 形式)(1 行につき 1 個の配列(列番号と列別データ))

キー{resultdata} -> キー{CONTENTS} -> キー{BODY} -> キー{(1 以降、該当レコードの存在行数を上限とする数値)}の中に、0 から始まる数値をキーとする

配列として格納されます。

表 2-4 レスポンスパラメーター覧(レコード情報)

列番号	列データ
0	一列目データ配列
1	二列目データ配列
⋮	⋮

Method：Get でレスポンスされるデータの表と、Json の階層構造を以下に示します。

表 2-5 返されるデーター覧

	0	1	2
0	A	I	U
1	あ	い	う
2	か	き	く
3	さ	し	す

▽JSON 形式

```
{
    "resultdata": {
        "CONTENTS": {
            " ├RECORD_LENGTH": 3,
            " └BODY": {
                "0": [
                    "A",
                    "I",
                    "U"
                ],
                "1": [
                    "あ",
                    "い",
                    "う"
                ],
                "2": [
                    "か",
                    "き",
                    "く"
                ],
                "3": [
                    "さ",
                    "し",
                    "す"
                ]
            }
        }
    }
}
```

(2) INFO(X-Command)

列情報のみを取得します。
X-Command(FILTER)または X-Command(EDIT)を実行する際に必要な情報を取得することができます。

・HTTP ヘッダ

表 2-6 HTTP ヘッダパラメーター覧

HTTP ヘッダ	値
Method	POST
X-Command	INFO

・content パラメータ

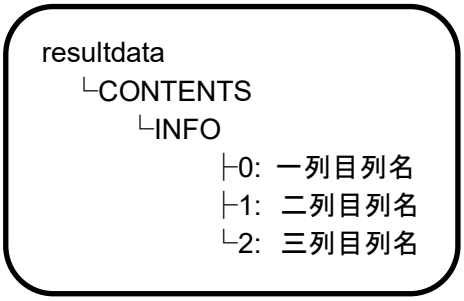
なし

・レスポンス

1)列情報（列番号と列名）

（JSON 形式）

キー{resultdata} -> キー{CONTENTS} -> キー{INFO}の中に、
0 から始まる数値をキーとする配列として格納されます。



(3) FILTER(X-Command)

パラメータで指定した条件に合致したレコードの、列情報（列番号と列名）および、
通常ステータス（廃止または活性中）の全レコードの行数とレコード内容を返却します。

表 2-7 HTTP ヘッダパラメーター覧

HTTP ヘッダ	値
Method	POST
X-Command	FILTER

・content パラメータ

1）フィルタ形式

操作したい Web ページの表示フィルタでフィルタをかけられる列と形式に従い、
列ごとに以下をフィルタの種類として指定可能です。

・NORMAL - 通常の LIKE 検索

・RANGE - 1~5 等の範囲検索

また、Web ページの表示フィルタにプルダウンが表示されている列については、
LIST（複数の完全一致条件による OR 検索。複数条件を配列に格納し指定）設定可能です。

2)指定形式

JSON 形式で指定。フィルタ種類ごとの形式で、フィルタ条件を格納します。
JSON 形式にする前段階では、1つの連想配列の中に、列番号ごとの連想配列を入れ子にする形で指定します。複数の列番号ごとの連想配列を格納した場合、AND で繋いだ意味になります。

さらに列番号ごとの連想配列の中にフィルタ条件の形式と条件をセットにした連想配列を格納してください。列番号ごとの連想配列に複数のフィルタ条件の連想配列を格納した場合、OR でつないだ意味になります。

・パラメータの記述例

例） FILTER パラメータの記述

列番号 2 が列名【項番】(主キーのカラム列)、列番号 4 が【備考】のコンテンツの場合で、
【項番】が 5 以上かつ、【備考】に「あいう」が入っているレコードを抽出したい場合

↓抽出イメージ

	列番号 1	列番号 2	列番号 3	列番号 4
列名	【列名 1】	【項番】	【列名 3】	【備考】
1	*****	1	*****	あいうえお
2	*****	2	*****	かきくけこ
3	*****	3	*****	あいうえお
4	*****	4	*****	かきくけこ
5	*****	5	*****	かきくけこ
6	*****	6	*****	あいうえお
⋮	⋮	⋮	⋮	⋮

▽JSON 形式

```
{
    "2": {
        "RANGE": {
            "START": 5
        }
    },
    "4": {
        '"NORMAL": "あいう"
    }
}
```

例） FILTER パラメータの記述(2)

列番号 2 が列名【項番】(主キーのカラム列)、のコンテンツの場合で、
【項番】が 10 から 99 の範囲、または、【項番】が 1 または 2 または 5、のレコードを抽出したい場合。

↓抽出イメージ

	列番号 1	列番 2	…
列名	【列名1】	【項番】	…
1	*****	1	…
2	*****	2	…
3	*****	3	…
4	*****	4	…
5	*****	5	…
⋮	⋮	⋮	…
10	*****	10	…
⋮	⋮	⋮	…
99	*****	99	…
⋮	⋮	⋮	…

▽JSON 形式

```
{
    "2": {
        "RANGE": {
            "START": "10",
            "END": "99"
        },
        "LIST": [
            "1",
            "2",
            "5"
        ]
    }
}
```

例） FILTER パラメータの記述(3)

列番号 2 が列名【項番】(主キーのカラム列)、列番号 5 が列名【最終更新日付】(日付型/日時型)のコンテンツの場合で、【項番】が 1 から 100 で、かつ、【最終更新日付】が 2016 年 8 月 01 日(00:00:00)から 2016 年 12 月 31 日(23:59:59)の範囲のレコードを抽出したい場合

▽JSON 形式

```
{
    "2": {
        "RANGE": {
            "START": "1",
            "END": "100"
        }
    },
    "5": {
        "RANGE": {
            "START": "2016/08/01 00:00:00",
```

```
                "END": "2016/12/31 23:59:59"
            }
        }
}
```

・レスポンス

1)レコード行数
（JSON 形式）
キー{resultdata} -> キー{CONTENTS} -> キー{RECORD_LENGTH}の中に、
数値として格納されます。

2)列情報（列番号と列名）
（JSON 形式）
キー{resultdata} -> キー{CONTENTS} -> キー{BODY} -> キー{0}の中に、
0 から始まる数値を、キーとする配列として格納される。

表 2-8 レスポンスパラメーター一覧(列情報)

列番号	列名
0	一列目
1	二列目
⋮	⋮

3)レコード情報
（JSON 形式）（1 行につき 1 個の配列（列番号と列別データ））
キー{resultdata} -> キー{CONTENTS} -> キー{BODY} -> キー{(1 以降、該当レコードの存在行数を上限とする数値)}の中に、0 から始まる数値をキーとする配列として格納される。

表 2-9 レスポンスパラメーター一覧(レコード情報)

列番号	列データ	説明
0	一列目データ	
1	二列目データ	
⋮	⋮	

※Json 格納時の階層構造は、Method:GET と同様です。

(4) EDIT(X-Command)

レコードの登録、既存レコードの更新、廃止、復活を行います。

・HTTP ヘッダ

表 2-10 HTTP ヘッダパラメーター一覧

HTTP ヘッダ	値
Method	POST
X-Command	EDIT

・パラメータ

1)指定形式
JSON 形式で指定してください。
INFO で取得できる列情報をもとに、1レコードにつき1つの配列で指定し、1レコードを格納した配列を要素とする配列を、JSON 形式でエンコードしたものを HTTP リクエスト

の context として送信してください。
列番号 0 の列名【処理種別】には、[登録]、[更新]、[廃止]、[復活]のいずれかを
指定してください。

例(1)　登録

列番号 0 が列名【処理種別】、列番 1 が【廃止】、列番号 2 が【項番】(主キー役のカラム列)、
・・・（中略）・・・
列番号 10 が【備考】、列番号 11 が【最終更新日時】、列番号 12 が【更新用の最終更新日時】、列番号 13 が【最終更新者】、のコンテンツの場合に、
2 レコードを追加する場合、

▽JSON 形式

```
{
    "0": {
        "0": "登録",
        "1": "",
        "2": "",
            ・・・（中略）・・・
        "10": "備考",
        "11": "",
        "12": "",
        "13": ""
    },
    "1": {
        "0": "登録",
        "1": "",
        "2": "",
            ・・・（中略）・・・
        "10": "備考",
        "11": "",
        "12": "",
        "13": ""
    }
}
```

例(2)　更新

列番号 0 が列名【処理種別】、列番号 1 が【廃止】、列番号 2 が【項番】(主キー役のカラム列)、
・・・（中略）・・・
列番号 9 が【備考】、列番号 10 が【最終更新日時】、
列番号 11 が【更新用の最終更新日時】、列番号 12 が【最終更新者】、のコンテンツの場合に、
【項番】10 のレコードを更新する場合、

▽JSON 形式

```
{
    "0": "更新",
```

```
        "1": "",
        "2": "10",
        "9": "備考欄",
        "10": "2016/08/01 12:30:45",
        "11": "【更新用の最終更新日時】",※
        "12": "管理者"
}
```

※Method：GET、X-Command：FILTER で取得した、【更新用の最終更新日時】をセットしてください。このデータによって、追い越し更新を防止しています。
【更新用の最終更新日時】は、“T_“で始まっています。

例(3)　登録（ファイルアップロードあり）

列番号 0 が列名【処理種別】、列番 1 が【廃止】、列番号 2 が【項番】(主キー役のカラム列)、・・・（中略）・・・
列番号 5 が【備考】、列番号 6 が【最終更新日時】、列番号 7 が【更新用の最終更新日時】、列番号 8 が【最終更新者】、のコンテンツの場合に、

▽JSON 形式

・　1 レコードを追加する場合、ファイルのアップロードあり

```
{
    "0": {
        "0": "登録",
        "3": "PV05004",
        "4": "20191226095004.yml",
        "5": "TEST"
    }
    "UPLOAD_FILE": [
        {
            "4": "<対象ファイルを base64encode をした値>"
        }
    ]
}
```

・　2 レコードを追加する場合、ファイルのアップロードあり

```
{
    "0": {
        "0": "登録",
        "3": "PV05004",
        "4": "20191226095004.yml",
        "5": "TEST"
    },
    "1": {
        "0": "登録",
        "3": "PV15004",
        "4": "20191226095004.yml",
        "5": "TEST"
    },
    "UPLOAD_FILE": [
```

```
            {
                "4": "<対象ファイルを base64encode をした値>"
            },
            {
                "4": "<対象ファイルを base64encode をした値>"
            }
        ]
}
```

※UPLOAD_FILE について、対象ファイルを base64encode をした値を指定し、ファイルのアップロードを行います。
※ファイルをアップロードする場合、要素順に「UPLOAD_FILE」に追加します。

・レスポンス

1)各レコードの処理結果

（JSON 形式）

キー{resultdata} -> キー{LIST} -> キー{NORMAL} -> キー{register、update、delete、error}の中に、配列として格納されます。

表 2-11 Key パラメーター覧

key	値の型	
name	文字列	処理結果種類の名前
ct	数値	（処理結果ごとの） レコード件数

2)各レコードの処理結果

（JSON 形式）

キー{resultdata} -> キー{LIST} -> キー{RAW} -> キー{パラメータとして渡したレコード番号（列情報を送信しなくてよい）デフォルト設定では、0、から始まる）}の中に、0 から始まる数値を、キーとする配列として格納されます。

表 2-12 Key パラメーター覧

key	値の型	
0	文字列	結果コード(別表を参照)
1	文字列	詳細コード(別表を参照)
2	文字列	エラーメッセージ

・レスポンス結果階層表

```
resultdata
  └LIST
      ├NORMAL
      |  ├register: {name:,ct:}
      |  ├update:  {name:,ct:}
      |  ├delete:   {name:,ct:}
      |  └error:    {name:,ct:}
      |
      └RAW
          ├0:{0:,1:,2:}
          ├1: {0:,1:,2:}
```

```
                    ├2: {0:,1:,2:}
                    ├ ・
                    ├ ・
                      ・
```

・レスポンス

返されるレスポンスには、JSON 形式で格納されています。

```
{
    "status": "SUCCEED",
    "resultdata": {
        "LIST": {
            "NORMAL": {
                "register": {
                    "name": "登録",
                    "ct":
                },
                "update": {
                    "name": "更新",
                    "ct":
                },
                "delete": {
                    "name": "廃止",
                    "ct":
                },
                "revive": {
                    "name": "復活",
                    "ct":
                },
                "error": {
                    "name": "エラー",
                    "ct":
                }
            },
            "RAW": [
                [
                    ・・・別表：結果コード／詳細コード　一覧・・・
                ],
                  ・・・（中略）・・・
            ]
        }
    }
}
```

別表:結果コード／詳細コード　一覧

処理種別	結果コード	詳細コード	説明
登録	000	201	登録の成功。
登録	002	000	必須項目が未入力。
登録	002	000	レコードと重複している項目がある。
登録	002	000	重複禁止に違反しているレコードが存在している。
登録	002	000	入力値の長さが規定のバイト数を超えている。
登録	002	000	入力値[NULL バイト文字等が含まれた値]が不正。
登録	002	000	半角整数以外が入力された。
登録	002	000	値が範囲外。
登録	002	000	入力された値が最小値を下回っているか最大値を上回っている。
登録	002	000	入力条件を満たしていない。
登録	002	000	数値以外が入力された。
登録	002	000	タブと改行が入力された。
登録	002	000	タブが入力された。
登録	002	000	入力値が範囲外。
登録	002	000	入力値が、PHP 関数(checkdate)で正常に処理できる範囲外。
登録	002	000	利用できない値が選択された。
登録	002	000	登録時に指定できない項目(主キー)が指定された。
登録	-	-	メンテナンス権限がない。
更新	000	200	更新の成功。
更新	002	000	必須項目が未入力。
更新	002	000	レコードと重複している項目がある。
更新	002	000	重複禁止に違反しているレコードが存在している。
更新	002	000	入力値の長さが規定のバイト数を超えている。
更新	002	000	入力値[NULL バイト文字等が含まれた値]が不正。
更新	002	000	半角整数以外が入力された。
更新	002	000	値が範囲外。
更新	002	000	入力された値が最小値を下回っているか最大値を上回っている。
更新	002	000	入力条件を満たしていない。
更新	002	000	数値以外が入力された。
更新	002	000	タブと改行が入力された。
更新	002	000	タブが入力された。
更新	002	000	入力値が範囲外。
更新	002	000	入力値が、PHP 関数(checkdate)で正常に処理できる範囲外。
更新	002	000	利用できない値が選択された。
更新	003	000	別セッションからレコードが更新されたため、更新の実行が中止された。
更新	003	000	廃止済レコードへの更新が実行されようとした。
更新	101	000	更新対象の行が特定できなかった。
更新	-	-	メンテナンス権限がない。
廃止	000	210	廃止の成功。
廃止	002	000	入力値の長さが規定のバイト数を超えている。
廃止	002	000	入力値[NULL バイト文字等が含まれた値]が不正。
廃止	002	000	入力された値が最小値を下回っているか最大値を上回っている。
廃止	002	000	入力条件を満たしていない。
廃止	002	000	タブが入力された。

処理種別	結果コード	詳細コード	説明
廃止	003	000	別セッションからレコードが更新されたため、廃止の実行が中止された。
廃止	003	000	廃止済レコードへの廃止が実行されようとした。
廃止	101	000	廃止対象の行が特定できなかった。
廃止	-	-	メンテナンス権限がない。
復活	000	200	復活の成功。
復活	002	000	必須項目が未入力。
復活	002	000	復活の場合、更新できない項目が更新されようとした。
復活	002	000	レコードと重複している項目がある。
復活	002	000	重複禁止に違反しているレコードが存在している。
復活	002	000	入力値の長さが規定のバイト数を超えている。
復活	002	000	入力値[NULL バイト文字等が含まれた値]が不正。
復活	002	000	入力された値が最小値を下回っているか最大値を上回っている。
復活	002	000	入力条件を満たしていない。
復活	002	000	タブが入力された。
復活	003	000	別セッションからレコードが更新されたため、復活の実行が中止された。
復活	003	000	復活済レコードへの復活が実行されようとした。
復活	101	000	復活対象の行が特定できなかった。
復活	-	-	メンテナンス権限がない。
表示	-	-	バリデーションエラー。
表示	-	-	次のいずれか(全レコード,廃止含まず,廃止のみ)が選択されていない。
-	000	000	処理スキップして、次のレコードへ。

3　メニューエクスポート/インポート利用編

3.1　メニューエクスポートを対象とした RestAPI

メニューエクスポートの操作を RestAPI で行うことができます。
利用可能な機能は、メニューグループ「ITA 基本コンソール」の、「メニューエクスポート」メニューに相当する操作です。

表 3-1 メニュー一覧

メニューグループ	メニュー名	メニューID
管理コンソール	メニューエクスポート	2100000211

3.1.1 リクエストの形式

下記の情報で HTTP リクエストを発行します。

・パス
　https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=メニューID
　メニューID は表 3-3　X-Command に指定可能なパラメーター覧　**を参照してください。**

・HTTP ヘッダ

表 3-2 HTTP ヘッダパラメーター覧

HTTP ヘッダ	説明
Method	POST のみ
Content-Type	“application/json”を指定する。
Authorization	ITA の認証要メニューにアクセスする場合は、 「ログイン ID」と「パスワード」* を、半角コロン(:)で結合して、base64encode をした値、を指定。
X-Command	EXECUTE INFO の 2 つが選択可能

X-Command に指定可能なパラメータ

表 3-3　**X-Command に指定可能なパラメーター覧**

X-Command	説明	対象画面	メニューID
INFO	エクスポート可能なメニューの一覧を取得します。	メニューエクスポート	2100000211
EXECUTE	エクスポートを実行します。	メニューエクスポート	2100000211

以下では、それぞれの X-comannd パラメータについての説明を行います。

3.1.2 INFO

エクスポート可能なメニューの一覧を出力します。

・パラメータ

指定するパラメータはありません。

・レスポンス

返されるレスポンスには、JSON 形式で格納されています。

```
{
    "status": "SUCCEED",
    "resultdata": {
        "MENU_LIST": {
            "メニューグループ ID ": {
                "menu_group_name": "メニューグループ名",
                "menu": [
                    {
                        "menu_id": "メニューID ",
                        "menu_name": "メニュー名"
                    },
                    {
                        ・・・（中略）・・・
                    }
                ]
            },
            "メニューグループ ID ": {
                        ・・・（中略）・・・
            }

        }
    }
}
```

表 3-4 レスポンス項目一覧

項目名	備考
メニューグループ ID	メニューグループ ID をキーとしてメニューの配列を構成する
menu_group_name	メニューグループ名
menu_id	メニューID
menu_name	メニュー名

3.1.3 EXECUTE

対象のメニューを指定し、エクスポートを実行します。

・パラメータ

以下を JSON 形式で content に指定してください。

表 3-5　メニューエクスポートパラメータ

パラメータ名	設定値
メニューグループ ID	メニューID

※メニューグループ ID、メニューID は、INFO の返り値で取得したものです。

例） JSON 記述

```
{
    "2100000002": [
        2100000202,
            ・・・（中略）・・・
        2100000222
    ],
    "2100000003": [
            ・・・（中略）・・・
    ],
}
```

・レスポンス

返されるレスポンスには、JSON 形式で格納されています。項目について以下参照。

```
{
    "status": "SUCCEED",
    "resultdata": {
        "TASK_ID": "実行 No ",
        "RESULTCODE": "結果コード",
        "RESULTINFO": "詳細情報"
    }
}
```

表 3-6 レスポンス項目一覧

項目名	備考
TASK_ID	作業 No
RESULTCODE	コマンド実行の成否のコード 000：正常終了 002：実行不可
RESULTINFO	詳細情報

3.2 メニューインポートを対象とした RestAPI

メニューエクスポートの操作を RestAPI で行うことができます。
利用可能な機能は、メニューグループ「ITA 基本コンソール」の、「メニューインポート」メニューに相当する操作です。

表 3-7 メニュー一覧

メニューグループ	メニュー名	メニューID
管理コンソール	メニューインポート	2100000212

3.2.1 リクエストの形式

下記の情報で HTTP リクエストを発行します。

・パス

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=メニューID
メニューID は 表 3-9 **X-Command に指定可能なパラメーター覧 を参照してください。**

・HTTP ヘッダ

表 3-8 **HTTP ヘッダパラメーター覧**

HTTP ヘッダ	説明
Method	POST のみ
Content-Type	“application/json”を指定する。
Authorization	ITA の認証要メニューにアクセスする場合は、 「ログイン ID」と「パスワード」* を、半角コロン(:)で結合して、base64encode をした値、を指定。
X-Command	UPLOAD EXECUTE の 2 つが選択可能

X-Command に指定可能なパラメータ

表 3-9 X-Command に指定可能なパラメーター覧

X-Command	説明	対象画面	メニューID
UPLOAD	エクスポートした kym ファイルのアップロードを実施し、インポート可能なメニューリストを出力します。	メニューインポート	2100000212
EXECUTE	インポート対象のメニューを選択し、インポートを実施する。	メニューインポート	2100000212

以下では、それぞれの X-comannd パラメータについての説明を行います。

3.2.2 UPLOAD

エクスポートされたファイルのアップロードを実行します。
ファイルは base64encode したものをパラメータとして転送します。

・パラメータ
以下を JSON 形式で content に指定してください。

表 3-10 **メニューインポート UPLOAD パラメーター覧**

パラメータ名	設定値
name	対象のファイル名
base64	対象のファイルを base64encode した値を指定

1）UPLOAD Json 記述例

```
{
   "zipfile":{
      "name":"ita_exportdata_20191224092830.kym",
      "base64":"・・・中略・・・"
   }
}
```

・レスポンス
返されるレスポンスには、JSON 形式で格納されています。

```
{
    "status": ""SUCCEED",
    "resultdata": {
        "upload_id": "アップロード ID ",※
        "data_portability_upload_file_name": "ファイル名",
        "IMPORT_LIST": {
            "メニューグループ ID ": {
                "menu_group_name": "メニューグループ名",
                "menu": [
                    {
                        "menu_id": "メニューID ",
                        "menu_name": "メニュー名"
                    }
                      ・・・（中略）・・・

                ],
            "メニューグループ ID ": {
                      ・・・（中略）・・・
                ],

                "RESULTCODE": "結果コード",
                "RESULTINFO": "詳細情報"
            }
```

```
        }
    }
}
```

※「upload_id」インポートの操作（EXECUTE）する際に使用します。

表 3-11 レスポンス項目一覧

項目名	備考
upload_id	アップロード成功時に付与される値 EXECUTE 時に使用
data_portability_upload_file_name	ファイル名
メニューグループ ID	メニューグループ ID をキーとしてメニューの配列を構成する
menu_group_name	メニューグループ名
menu_id	メニューID
menu_name	メニュー名
RESULTCODE	コマンド実行の成否のコード 000：正常終了 002：実行不可
RESULTINFO	詳細情報

3.2.3 EXECUTE

アップロードしたファイルを元に、インポート処理を実行します。
対象とする、メニューグループ、メニューID、インポート実行モードを指定できます。

・パラメータ
以下を JSON 形式で content に指定してください。

表 3-12 **メニューインポート EXECUTE パラメータ**

パラメータ名	設定値	備考
メニューグループ ID	メニューID	
importButton	空	インポート時指定にキー指定
importButton2	空	インポート（廃止を除く）時、キー指定
upload_id		UPLOAD の返り値で取得したものの先頭いに”A_”を付与した値
data_portability_upload_file_name	ファイル名	

1）UPLOAD Json 記述例

```
{
    "2100070001": [
        2100070001,
        2100070002,
        2100070003
    ],
    "2100020002": [
        ・・・（中略）・・・
```

```
        ],

        "importButton": "",
        "upload_id": "A_2019121709033577204023 9",※
        "data_portability_upload_file_name": "ita_exportdata_20191213095733.kym"
}
```

※UPLOAD で取得した「upload_id」の先頭に”A_”を付与して使用します。

・レスポンス

返されるレスポンスには、JSON 形式で格納されています。項目について以下参照。

```
{
    "status": "SUCCEED",
    "resultdata": {
        "TASK_ID": "メニューインポート実行の作業 No",
        "RESULTCODE": "結果コード",
        "RESULTINFO": "詳細情報"
    }
}
```

表 3-13 レスポンス項目一覧

項目名	備考
TASK_ID	作業 No
RESULTCODE	コマンド実行の成否のコード 000:正常終了 002:実行不可
RESULTINFO	詳細情報

4 Symphony 利用編

4.1 Symphony 登録作業を対象とした RestAPI

Symphony の操作を RestAPI で行うことができます。
利用可能な機能は、メニューグループ「ITA 基本コンソール」の、「Symphony クラス編集」メニューに相当する操作です。

表 4-1 対象メニュー一覧

メニューグループ	メニュー名	メニューID
基本コンソール	Symphony 編集	2100000306

リクエスト

4.1.1 リクエストの形式

下記の情報で HTTP リクエストを発行します。

・パス

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=メニューID
メニューID は**表 4-3 X-Command に指定可能なパラメーター覧 を参照してください。**

・HTTP ヘッダ

表 4-2 HTTP ヘッダパラメーター覧

HTTP ヘッダ	説明
Method	POST のみ
Content-Type	“application/json”を指定する。
Authorization	ITA の認証要メニューにアクセスする場合は、 「ログイン ID」と「パスワード」* を、半角コロン(:)で結合して、base64encode をした値、を指定。
X-Command	INFO FILTER EDIT の 3 つが選択可能

X-Command に指定可能なパラメータ

表 4-3 X-Command に指定可能なパラメーター覧

X-Command	説明	対象画面	メニューID
INFO	Symphony クラスの列情報のみを取得します。	Symphony 編集	2100000306
FILTER	Symphony クラスのパラメータに一致したレコードの参照を行います。	Symphony 編集	2100000306
EDIT	Symphony クラスの登録を行います。	Symphony 編集	2100000306

以下では、それぞれの X-comannd パラメータについての説明を行います。

4.1.2 INFO

Symphony クラスの列情報を取得します。
※詳細は「標準 REST 機能の利用」-「INFO(X-Command)」を参照

4.1.3 FILTER

パラメータで指定した条件に合致したレコードの、列情報（列番号と列名）および、
通常ステータス（廃止または活性中）の全レコードの行数とレコード内容と列情報を取得します。
※詳細は「標準 REST 機能の利用」-「FILTER(X-Command)」を参照

4.1.4 EDIT

Symphony クラスの登録、更新、廃止、復活を行います。

・HTTP ヘッダ

表 4-4 HTTP ヘッダパラメーター覧

HTTP ヘッダ	値
Method	POST
X-Command	EDIT

・パラメータ

1) 指定形式

各実行種別のパラメータ指定項目は以下の、パラメータ指定項目を参照してください。
項目番号 7 について、[更新]、[廃止]、[復活]指定時、
X-Command：FILTER で取得した、【更新用の最終更新日時】をセットしてください。
このデータによって、追い越し更新を防止しています。
【更新用の最終更新日時】は、“T_“で始まっています。

表 4-5 Symphony **クラスパラメーター覧**

項目番号	パラメータ名	備考
0	実行処理種別	登録/更新/廃止/復活
2	Symphony クラス ID	登録時は、空で実施。
3	Symphony 名称	
4	説明	
5	備考	
7	更新用の最終更新日時	T_XXXXXXXXXXXXXXXXXXX
9	Movement 詳細	Movement 情報 詳細は以下、表 4-6

表 4-6 Movement 詳細**一覧**

項目番号	パラメータ名	備考
0	Orchestrator ID	オーケストレータの ID 対応表は以下表 4-8
1	Movement ID	Movement の ID 「Movement 一覧」メニュー参照

2	一時停止	OFF:空 ON:checkedValue
3	説明	
4	オペレーション ID(個別指定)	オペレーションの ID 「投入オペレーション」メニューを参照。

表 4-7 Symphony **クラスパラメータ** Movement 詳細

パラメータ指定項目（登録/更新）

```
"9": [
    {
        "0": "Orchestrator ID",
        "1": "Movement ID",
        "2": "一時停止(OFF:/ON:checkedValue)",
        "3": "説明",
        "4": "オペレーション ID(個別指定)"
    },
    {
     ・・・複数の Movement 実行の場合、実行順に追加・・・・
    }

]
```

表 4-8 オーケストレータの ID 対応表

ID	ステータス
3	Ansible Legacy
4	Ansible Pioneer
5	Ansible Legacy Role
8	DSC
9	OpenStack

パラメータ指定項目（登録/更新）

```
{
    "0": "実行処理種別：<登録　or　更新>",
    "2": "Symphony クラス ID ",
    "3": "Symphony 名称",
    "4": "説明",
    "5": "備考",
    "7": "更新用の最終更新日時",
    "9": [
        {
            "0": "Orchestrator ID",
            "1": "Movement ID",
            "2": "一時停止(OFF:/ON:checkedValue)",
            "3": "説明",
            "4": "オペレーション ID(個別指定)"
            }
     ・・・複数の場合、Movement の実行順で繰り返し追加・・・・
```

```
    ]
}
```

パラメータ指定項目（廃止/復活）

```
{
    "0": "実行処理種別：<廃止　or 復活>",
    "2": "Symphony クラス ID",
    "7": "更新用の最終更新日時"
}
```

※Method：GET、X-Command：FILTER で取得した、【更新用の最終更新日時】をセットしてください。このデータによって、追い越し更新を防止しています。
【更新用の最終更新日時】は、“T_“で始まっています。

例）JSON 記述例：複数の実行処理種別対象とした場合

```
[
    {
        "0": "登録",
        "2": "",
        "3": "DEMO_001_20191224135448_0",
        "4": "demo_001_20191224135448_0",
        "7": "",
        "9": [
            {
                "1": 3,
                "2": 1,
                "3": "checkedValue",
                "4": "DEMO_MOVE_0",
                "5": 1
            },
            {
                "1": 3,
                "2": 2,
                "3": "",
                "4": "DEMO_MOVE_1",
                "5": ""
            }
        ]
    },
    {
        "0": "更新",
        "2": 1,
        "3": "DEMO_001_20191224135448_1",
        "4": "demo_001_20191224135448_1",
        "7": "T_20191224113132971799",
        "9": [
```

```
                {
                    "1": 3,
                    "2": 1,
                    "3": "",
                    "4": "DEMO_MOVE_0",
                    "5": 1
                }
            ]
        },
        {
            "0": "廃止",
            "2": 2,
            "7": "T_20191224135437197447"
        },
        {
            "0": "復活",
            "2": 4,
            "7": "T_20191224135449793941"
        }
]
```


・レスポンス

各レコードの処理結果について、「標準 REST 機能の利用」-「EDIT(X-Command)」を参照。

4.2 Symphony 作業実行を対象とした RestAPI

Symphony の操作を RestAPI で行うことができます。
利用可能な機能は、メニューグループ「ITA 基本コンソール」の、「Symphony 作業実行」メニュー、「Symphony 作業確認」メニューに相当する操作です。

表 4-9 対象メニュー一覧

メニューグループ	メニュー名	メニューID
基本コンソール	Symphony 作業実行	2100000309

4.2.5 リクエストの形式

下記の情報で HTTP リクエストを発行します。

・パス

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=メニューID
メニューID は **表 4-11 X-Command に指定可能なパラメーター覧を参照してください。**

・HTTP ヘッダ

表 4-10 HTTP ヘッダパラメーター覧

HTTP ヘッダ	説明
Method	POST のみ
Content-Type	“application/json”を指定する。
Authorization	ITA の認証要メニューにアクセスする場合は、 「ログイン ID」と「パスワード」* を、半角コロン(:)で結合して、base64encode をした値、を指定。
X-Command	EXECUTE CANCEL SCRAM RELEASE の 4 つが選択可能

X-Command に指定可能なパラメータ

表 4-11 X-Command に指定可能なパラメーター覧

X-Command	説明	対象画面	メニューID
EXECUTE	Symphony の作業実行を行います。	Symphony 作業実行	2100000308
CANCEL	Symphony の予約取り消しを行います。	Symphony 作業確認	2100000309
SCRAM	Symphony の緊急停止を行います。	Symphony 作業確認	2100000309
RELEASE	Symphony の一時停止ポイントの解除を行います。	Symphony 作業確認	2100000309

以下では、それぞれの X-comannd パラメータについての説明を行います。

4.2.6 レスポンスの項目

以下では、それぞれの X-comannd 実行時のレスポンス項目についての説明を行います。

表 4-12 レスポンス項目一覧

項目名	備考
SYMPHONY_INSTANCE_ID	SYMPHONY インスタンスに対する操作に使用
MOVEMENT_SEQ_NO	RELEASE 時のみ使用
RESULTCODE	コマンド実行の成否のコード 000:正常終了 001:実行不可 002:予約取消不可 003:緊急停止不可 004:一時停止解除不可
RESULTINFO	詳細情報

4.2.7 EXECUTE

Symphony クラスとオペレーションを指定して、作業実行を行います。予約日時の指定や、Symphony クラスに登録されている Movement ごとに、スキップ、オペレーション ID の個別指定ができます。

・パラメータ

以下を JSON 形式で content に指定してください。

表 4-13 オペレーション ID 個別指定パラメーター覧

パラメータ名	設定値
SYMPHONY_CLASS_NO	Symphony クラス ID
OPERATION_ID	オペレーション ID
PRESERVE_DATETIME	予約日時（YYYY/MM/DD tt:mm）
OPTION	スキップの有無、オペレーション ID の個別指定の配列

・OPTION の指定

OPTION には、配列で Movement ごとに、スキップ、オペレーション ID の個別指定ができます。

・Movement 要素の階層

```
├1（Movement の実行順番）
|       ├SKIP   -   YES or NO
|       └OPERATION_ID   -   (個別指定するオペレーション ID)
├2（Movement の実行順番）
|       ├SKIP   -   YES or NO
|       └OPERATION_ID   -   (個別指定するオペレーション ID)
・
・
```

1）EXECUTE　Json 記述例

Symphony クラス ID が 1、オペレーション ID が 1001、予約日時が 2016/01/01 00:00 の場合
さらに、1 番目に実行される Movement をスキップし、2 番目に実行される Movement のオペレーション ID に 2001 を指定
▽Json 形式で記述

```
{
    "SYMPHONY_CLASS_NO": 1,
    "OPERATION_ID": 1001,
    "PRESERVE_DATETIME": "2016/01/0100:00",
    "OPTION": {
        "1": {
            "SKIP": "YES"
        },
        "2": {
            "OPERATION_ID": 2001
        }
    }
}
```

図　6.1-1 EXECUTE　Json **記述例**

・レスポンス
返されるレスポンスには、JSON 形式で格納されています。

```
{
    "status": "実行の成否",
    "resultdata": {
        "SYMPHONY_INSTANCE_ID": "実行 No ",※
        "RESULTCODE": "結果コード",
        "RESULTINFO": "詳細情報"
    }
}
```

※実行後のインスタンスを操作（INFO,CANCEL、SCRAM、RELEASE）する際に使用します。

4.2.8 CANCEL

予約日時が登録されている Symphony のインスタンス ID を指定して、予約実行をキャンセルします。

・パラメータ
以下を JSON 形式で content に指定してください。

表　4-14　Symphony 実行予約キャンセルパラメータ表

パラメータ名	設定値
SYMPHONY_INSTANCE_ID	Symphony インスタンス ID※

※EXECUTE の返り値で取得したものです。

・レスポンス
返されるレスポンスには、JSON 形式で格納されています。項目について以下参照。

```
{
    "status": "SUCCEED",
    "resultdata": {
        " SYMPHONY_INSTANCE_ID": " Symphony 実行時のインスタンスの ID ",
        "RESULTCODE": "結果コード",
        "RESULTINFO": "詳細情報"
    }
}
```

4.2.9 SCRAM

実行されている Symphony のインスタンス ID を指定して、緊急停止します。

・パラメータ

以下を JSON 形式で content に指定してください。

表 4-15 Symphony 実行処理の緊急停止パラメータ表

パラメータ名	設定値
SYMPHONY_INSTANCE_ID	Symphony インスタンス ID※

※EXECUTE の返り値で取得したものです。

・レスポンス

返されるレスポンスには、JSON 形式で格納されています。項目について以下参照。

```
{
    "status": "SUCCEED",
    "resultdata": {
        " SYMPHONY_INSTANCE_ID": " Symphony 実行時のインスタンスの ID ",
        "RESULTCODE": "結果コード",
        "RESULTINFO": "詳細情報"
    }
}
```

4.2.10 RELEASE

Symphony のインスタンス ID と Movement の順番を指定して、一時停止が設定されているポイントを解除します。

・パラメータ

以下を JSON 形式で content に指定してください。

表 4-16 Symphony 実行処理の一時停止解除パラメータ表

パラメータ名	設定値
SYMPHONY_INSTANCE_ID	Symphony インスタンス ID※
MOVEMENT_SEQ_NO	何番目の Movement か

※EXECUTE の返り値で取得したものです。

・レスポンス
返されるレスポンスには、JSON 形式で格納されています。項目について以下参照。

```
{
    "status": "SUCCEED",
    "resultdata": {
        "SYMPHONY_INSTANCE_ID": "Symphony 実行時のインスタンスの ID",
        "MOVEMENT_SEQ_NO": "実行した Movement の Symphony クラス内の順番",
        "RESULTCODE": "結果コード",
        "RESULTINFO": "詳細情報"
    }
}
```

4.3 Symphony 作業確認を対象とした RestAPI

Symphony の操作を RestAPI で行うことができます。
利用可能な機能は、メニューグループ「ITA 基本コンソール」の、「Symphony 作業実行」メニュー、「Symphony 作業確認」メニューに相当する操作です。

表 4-17 対象メニュー一覧

メニューグループ	メニュー名	メニューID
基本コンソール	Symphony 作業実行	2100000309

4.3.11 リクエストの形式

下記の情報で HTTP リクエストを発行します。

・パス
https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=メニューID
メニューID は 表 4-19 X-Command に指定可能なパラメーター覧**を参照してください。**

・HTTP ヘッダ

表 4-18 HTTP ヘッダパラメーター覧

HTTP ヘッダ	説明
Method	POST のみ
Content-Type	“application/json”を指定する。
Authorization	ITA の認証要メニューにアクセスする場合は、 「ログイン ID」と「パスワード」* を、半角コロン(:)で結合して、base64encode をした値、を指定。
X-Command	INFO の 1 つが選択可能

X-Command に指定可能なパラメータ

表 4-19 X-Command に指定可能なパラメーター覧

X-Command	説明	対象画面	メニューID

INFO	Symphony の状態確認をし、ステータスを返します。	Symphony 作業確認	2100000309

以下では、それぞれの X-comannd パラメータについての説明を行います。

4.3.12 レスポンスの項目

以下では、それぞれの X-comannd 実行時のレスポンス項目についての説明を行います。

表 4-20 レスポンス項目一覧

項目名	備考
SYMPHONY_INSTANCE_ID	SYMPHONY インスタンスに対する操作に使用
RESULTCODE	コマンド実行の成否のコード 000:正常終了
RESULTINFO	詳細情報

4.3.13 INFO

Symphony 実行時のインスタンス ID を指定して、実行時の情報を取得します。

・パラメータ

以下を JSON 形式で content に指定してください。

表 4-21 Symphony 実行情報取得パラメータ表

パラメータ名	設定値
SYMPHONY_INSTANCE_ID	Symphony インスタンス ID※

※EXECUTE の返り値で取得したものです。

・レスポンス

返されるレスポンスには、JSON 形式で格納されています。

```
{
    "status": "SUCCEED",
    "resultdata": {
        "SYMPHONY_CLASS_ID": "1",
        "SYMPHONY_INSTANCE_INFO": {
            "SYMPHONY_INSTANCE_ID": 1,
                ・・・（中略）項目は以下①参照・・・
            "FOCUS_MOVEMENT": 1
        },
        "MOVEMENTS": [
            {
                "CLASS_ITEM": {
                    "ORCHESTRATOR_ID": "3",
                        ・・・（中略）項目は以下②参照・・・
                    "NEXT_PENDING": "checkedValue"
                },
                "INS_ITEM": {
                    "STATUS": "11",
                        ・・・（中略）項目は以下③参照・・・
                    "OPERATION_NAME": null
```

```
                    }
                }
                 ・・・・
            ],
            "RESULTCODE": "000",
            "RESULTINFO": ""
        }
}
```


①SYMPHONY_INSTANCE_INFO に格納される Symphony インスタンスの情報配列

表 4-22 インスタンス配列表

キー	内容
SYMPHONY_INSTANCE_ID	Symphony インスタンス ID
I_SYMPHONY_CLASS_NO	このインスタンスの元クラスの ID
I_SYMPHONY_NAME	このインスタンスの元クラスの名前
I_DESCRIPTION	このインスタンスの元クラスの説明
STATUS_ID	実行時ステータス　詳細は以下　表**エラー! 参照元が見つかりません。**
ABORT_EXECUTE_FLAG	緊急停止発令フラグ　未発令:1　発令済み:2
OPERATION_NO_UAPK	登録オペレーション NO
OPERATION_NO_IDBH	登録オペレーション ID
OPERATION_NAME	登録オペレーション名
TIME_BOOK	予約日時
TIME_START	開始日時
TIME_END	終了日時
MOVEMENT_LENGTH	登録 Movement の数
FOCUS_MOVEMENT	現在の Movement は何番目か

②CLASS_ITEM に格納される Movement のクラス情報

表 4-23 Movement クラス情報表

キー	内容
ORCHESTRATOR_ID	オーケストレータの ID　対応表は以下　**エラー! 参照元が見つかりません。**
PATTERN_ID	Movement の ID
PATTERN_NAME	Movement の名前
THEME_COLOR	<画面用>Web 画面で設定時の○アイコンの色
MOVEMENT_SEQ	Symphony クラスの中で何番目か
DESCRIPTION	Symphony クラス編集画面で入力したコメント
NEXT_PENDING	一時停止が設定されている:checkedValue

③INS_ITEM に格納される Movement のインスタンス情報

表 4-24 Movement インスタンス情報表

キー	内容
STATUS	実行時ステータス　詳細は以下　**エラー! 参照元が見つかりません。**
RELEASED	一時停止が設定されている:1 一時停止解除された:2
EXECUTION_NO	Movement インスタンス ID
JUMP	<画面用>遷移先 URL
ABORT_RECEPTED	緊急停止を　1:受け付けていない　2:受付済み
SKIP	スキップが設定されている　:1
TIME_START	開始日時
TIME_END	終了日時
OPERATION_ID	個別指定されたオペレーション ID
OPERATION_NAME	個別指定されたオペレーション名

表 4-25 Symphony インスタンスの実行時ステータス ID 対応表

ID	ステータス
1	未実行
2	未実行(予約)
3	実行中
4	実行中(遅延)
5	正常終了
6	緊急停止
7	異常終了
8	想定外エラー
9	予約取消

表 4-26 オーケストレータの ID 対応表

ID	ステータス
3	Ansible Legacy
4	Ansible Pioneer
5	Ansible Legacy Role
8	DSC
9	OpenStack

表 4-27 Movement インスタンスの実行時ステータス ID 対応表

ID	ステータス
1	未実行
2	準備中
3	実行中
4	実行中(遅延)

5	実行完了
6	異常終了
7	緊急停止
8	保留中
9	正常終了
10	準備エラー
11	想定外エラー
12	Skip 完了
13	Skip 後保留中
14	Skip 終了

5 Movement 利用編

5.1 Movement 作業実行を対象とした RestAPI

Movement の操作を RestAPI で行うことができます。
利用可能な機能は、以下メニューグループに該当する「作業実行」、「作業状態確認」に相当する操作です。

表 5-1 作業実行、状態確認メニュー一覧

メニューグループ	メニュー名	メニューID
Ansible-Legacy	作業実行	2100020111
	作業状態確認	2100020112
Ansible-Pioneer	作業実行	2100020211
	作業状態確認	2100020212
Ansible-LegacyRole	作業実行	2100020312
	作業状態確認	2100020313
DSC	作業実行	2100060009
	作業状態確認	2100060010
OpenStack	作業実行	2100070004
	作業状態確認	2100070005

5.1.1 リクエストの形式

下記の情報で HTTP リクエストを発行します。

・パス

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=メニューID
メニューID は**表 6.1-2 X-Command に指定可能なパラメーター覧を参照してください。**

・HTTP ヘッダ

表 5-2 HTTP ヘッダパラメーター覧

HTTP ヘッダ	説明
Method	POST のみ
Content-Type	“application/json”を指定する。
Authorization	ITA の認証要メニューにアクセスする場合は、 「ログイン ID」と「パスワード」* を、半角コロン(:)で結合して、base64encode をした値、を指定。
X-Command	EXECUTE CANCEL SCRAM の 3 つが選択可能

X-Command に指定可能なパラメータ

表 5-3 X-Command に指定可能なパラメーター覧

X-Command	説明	対象画面	メニューID
EXECUTE	予約/作業実行を行います。	作業実行	2100020111 2100020211 2100020312 2100060009 2100070004
CANCEL	予約取り消しを行います。	作業状態確認	2100020112 2100020212 2100020313 2100060010 2100070005
SCRAM	緊急停止を行います。	作業状態確認	

以下では、それぞれの X-comannd パラメータについての説明を行います。

5.1.2 レスポンスの項目

以下では、それぞれの X-comannd 実行時のレスポンス項目についての説明を行います。

表 5-4 レスポンス項目一覧

項目名	備考
EXECUTION_NO	作業 No に対する操作に使用
RESULTCODE	コマンド実行の成否のコード 000:正常終了 001:実行不可 002:予約取消不可 003:緊急停止不可
RESULTINFO	詳細情報

5.1.3 EXECUTE

Movement クラスとオペレーションを指定して、作業実行を行います。予約日時の指定や、実行モード(ドライラン/実行)を指定できます。
ができます。

・パラメータ

以下を JSON 形式で content に指定してください。

表 5-5 Movement 実行パラメーター覧

パラメータ名	設定値
MOVEMENT_CLASS_ID	Movement クラス ID
OPERATION_ID	オペレーション ID
PRESERVE_DATETIME	予約日時(YYYY/MM/DD tt:mm)
RUN_MODE	1:実行、 2:ドライラン

例) JSON 記述例

```
{
    "MOVEMENT_CLASS_ID": 1,
```

```
    "OPERATION_ID": 1,
    "PRESERVE_DATETIME": "2019/12/24 15:44",
    "RUN_MODE": 1
}
```

・レスポンス
返されるレスポンスには、JSON 形式で格納されています。

```
{
    "status": "SUCCEED",
    "resultdata": {
        "EXECUTION_NO": "作業 No",
        "RESULTCODE": "結果コード",
        "RESULTINFO": "詳細情報"
    }
}
```

5.1.4 CANCEL

予約日時が登録されている作業 No を指定して、予約実行をキャンセルします。

・パラメータ
以下を JSON 形式で content に指定してください。

表 5-6 Movement 実行パラメーター覧

パラメータ名	設定値
EXECUTION_NO	作業 No※

※EXECUTE の返り値で取得したものです。

・レスポンス
返されるレスポンスには、JSON 形式で格納されています。

```
{
    "status": "SUCCEED",
    "resultdata": {
        "EXECUTION_NO": "作業 No",
        "RESULTCODE": "結果コード",
        "RESULTINFO": "詳細情報"
    }
}
```

5.1.5 SCRAM

実行されている作業 No を指定して、緊急停止します。

・パラメータ
以下を JSON 形式で content に指定してください。

表 5-7 Movement 実行パラメーター覧

パラメータ名	設定値
EXECUTION_NO	作業 No※

※EXECUTE の返り値で取得したものです。

・レスポンス
返されるレスポンスには、JSON 形式で格納されています。

```
{
    "status": "SUCCEED",
    "resultdata": {
        "EXECUTION_NO": "作業 No",
        "RESULTCODE": "結果コード",
        "RESULTINFO": "詳細情報"
    }
}
```