ITA_User Instruction Manual

RestAPI

— Version 1.3 —

※「Exastro IT Automation」is written as「ITA」 in this document.

1 Table of contents

2 Introduction

This document explains the function and the operation method of REST API in ITA system.

1 ITA system Overview of REST API

This chapter explains the standard REST API for operating ITA.

1.1 About REST API

ITA provides REST API that can perform various operations to resources that are managed in ITA from external programs.

- Standard RESTAPI can be used in the menu other than the menu described in the following table. Please refer to "2 Standard REST function" for the details of standard REST API.

Table 1-1 List of Individual REST API

Menu group	Menu name	Menu ID	Reference
Management console	Export menu	2100000211	3 Menu export / import
	Import menu	2100000212	
Basic console	Export Symphony/Operation	2100000401	4 Export･Import Symphony/Operation
	Import Symphony/Operation	2100000402	
	Symphony class editor	2100000306	5 Symphony
	Symphony execution	2100000308	
	Symphony execution checking	2100000309	
Ansible-Legacy	Execution	2100020111	6 Movement
	Check operation status	2100020112	
Ansible-Pioneer	Execution	2100020211	
	Check operation status	2100020212	
Ansible-LegacyRole	Execution	2100020312	
	Check operation status	2100020313	
DSC	Execution	2100060009	
	Check operation status	2100060010	
OpenStack	Execution	2100070004	
	Check operation status	2100070005	

2 Standard REST function

Using REST API from external programs to operate the resources managed in ITA is possible.
The following shows the calling convention.

2.1 Format of request

ITA REST API sends HTTP request to the path of each menu on ITA.

Path
https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=(Menu ID of each menu）
e.g.) In the case of "Management console" – "system settings" menu (Menu ID:2100000202)
https:// exastro-it-automation:443/default/menu/07_rest_api_ver1.php?no=2100000202

※<HostName>: The host name "exastro-it-automation" when installing ITA with ITA installer.

HTTP Header：
The items in the following table can be used.

Table 2-1 HTTP header parameter list

HTTP Header	Description
Host	Specify the host name of ITA RestAPI server or the IP address and port separated with colon (:)
Method	In principle：Specify POST Exception：GET is able to be specified when accessing the menu that doesn't need authentication on ITA.
Content-Type	Specify "application/json" It's optional if the Method is GET.
Authorization	When accessing the menu that needs ITA authentication, specify the value of "Login ID" and "Password"* connected with half-width colon (:) then encoded with base64. It's optional if the Method is GET.
X-Command	Can be set only when Method is POST. One of【INFO】､【FILTER】､【EDIT】can be set.

Whether the HTTP request is in uppercase or lowercase does not matter.

* If the ITA password has expired, RestAPI will turn out to be error.
Please perform request after changing the password from the login screen of Web system.
However, please follow the authentication information managed in ActiveDirectory when using the ActiveDirectory association function. (This limitation doesn't apply to the Non-association target user of ActiveDirectory association function.)
Please refer to "User Instruction Manual_Management console" – "Usage of AcitveDirectory association function" for the details of ActiveDirectory association function.

Example of HTTP header：

In the case that Login ID is [test_loginid] and password is [test_password]

```
Encrypt test_loginid: test_password　with base64 encoding
  →[qTImqS9fo2qcozyxBaEyp3EspTSmp3qipzD=]）
Host:<HostName>:<Port>
Content-Type:application/json
Authorization: qTImqS9fo2qcozyxBaEyp3EspTSmp3qipzD=
X-Command: INFO
```

2.2 Available Method and Command

The hierarchy of available method and command are as follows.

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

(1) GET (Method)
Column information (column number and column name) and the record line count and record content in normal state (discarded or active) will be returned.

・HTTP Header

Table 2-2 HTTP header parameter

HTTP Header	Value
Method	GET

・content parameter
None

・Response

1) Record line count
(JSON format)
The numeric value is stored key{resultdata} -> key{CONTENTS} -> key{RECORD_LENGTH}

2) Column information (Column number and column name)
(JSON format)
Stored as array with numeric key value starting from 0 in key{resultdata} -> key{CONTENTS} -> key{BODY} -> key{0}

Table 2-3 Response parameter list (Column information)

Column number	Column name
0	First column
1	Second column
⋮	⋮

3) Record information
(JSON format)(One array per row (column number and column-specific data))
Stored as array with numeric key value starting from 0 in key{resultdata} -> key{CONTENTS} -> key{BODY} -> key{(Numeric value starts from 1 to the maximum line count

of the according record)}

Table 2-4 List of Response parameter (Record information)

Column number	Column data
0	First data array
1	Second data array
⋮	⋮

The table of data reponsed by Method:GET and the Json hierarchy structure is as below

Table 2-5 List of returned data

	0	1	2
0	A	B	C
1	D	E	F
2	G	H	I
3	J	K	L

▽JSON format
```
{
        "resultdata": {
                "CONTENTS": {
                        " ├RECORD_LENGTH": 3,
                        " └BODY": {
                                "0": [
                                        "A",
                                        "B",
                                        "C"
                                ],
                                "1": [
                                        "D",
                                        "E",
                                        "F"
                                ],
                                "2": [
                                        "G",
                                        "H",
                                        "I"
                                ],
                                "3": [
                                        "J",
                                        "K",
                                        "L"
                                ]
                        }
                }
        }
}
```

(2) INFO(X-Command)

Obtain column information only.

It is possible to obtain required information only by executing X-Command (FILTER) or X-Command (EDIT).

・HTTP Header

Table 2-6 HTTP Header parameter list

HTTP header	Value
Method	POST
X-Command	INFO

・content parameter

None

・Response

1) Column information (Column number and column name)

(JSON format)

Stored as array with numeric key value starting from 0 in key{resultdata} -> key{CONTENTS} -> key{INFO}

```
resultdata
    └CONTENTS
        └INFO
              ├0: 1st column name
              ├1: 2nd column name
              └2: 3rd column name
```

(3) FILTER(X-Command)

The Column information (Column number and column name) and the line count and record content of the whole record in normal status (discarded or active) that meets the criteria specified in parameter is returned.

Table 2-7 HTTP header parameter list

HTTP header	Value
Method	POST
X-Command	FILTER

・content parameter

1) Filter format

By follow the format of the column that can be filtered by the display filter on the Web console, the following types of filters can be specified for each column.

・NORMAL - Normal LIKE search

・RANGE - Range search from level 1~5

Also, if the column is displayed in pulldown menu in the display filter on web console, specifying LIST ("OR" search with multiple exact match conditions. Specify multiple conditions in array) is available.

2) Specification format

Specify in JSON format. Store the filter criteria according to the format of filter type. When turning the data into JSON format, specify associative arrays for each column nested in one associative array. If multiple columns are stored in separated associative arrays, it means they are connected with AND relationship.

In addition, please set the format and criteria of filter criteria and store it in an associate array then put it in the associative array of each column. If multiple associative array of filter conditions is stored in single associative array, it means that they are connected with OR relationship.

・Parameter specification example

e.g.) Describing FILTER parameter

Column 2 is "ID" (primary key column), Column 4 is the content of "Remarks".
In the case of extracting the record whose "number" is 5 or more, and having "ABC" in the "Remarks"
↓Extraction image

	Column1	Column2	Column	Column
row	Column1	ID	Coumn3	Remarks
1	*****	1	*****	ABCDE
2	*****	2	*****	GHIJK
3	*****	3	*****	ABCDE
4	*****	4	*****	GHIJK
5	*****	5	*****	GHIJK
6	*****	6	*****	ABCDE
⋮	⋮	⋮	⋮	⋮

▽JSON format
```
{
    "2": {
        "RANGE": {
            "START": 5
        }
    },
    "4": {
        "'NORMAL": "ABC"
    }
}
```

e.g.)　Describing FILTER parameter (2)

Column 2 is "ID" (primary key column).
In the case of extracting the record whose "number" is from 10~99 or is 1,2,or 5.

↓Extraction image

	Column1	Column2	…
Row	Column1	ID	…
1	*****	1	…
2	*****	2	…
3	*****	3	…
4	*****	4	…
5	*****	5	…
⋮	⋮	⋮	…
10	*****	10	…
⋮	⋮	⋮	…
99	*****	99	…
⋮	⋮	⋮	…

▽JSON format

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

e.g.)　Describing FILTER parameter (2)

Column 2 is "ID" (primary key column), Column 5 is "last update date"(date type/date time type).
In the case of extracting the record whose "ID" is in 1 to 100 and the last update time is in 2016/8/1(00:00:00) to 2016/12/31(23:59:59)

▽JSON format

```
{
    "2": {
        "RANGE": {
            "START": "1",
            "END": "100"
        }
    },
    "5": {
        "RANGE": {
            "START": "2016/08/01 00:00:00",
            "END": "2016/12/31 23:59:59"
            }
        }
}
```

・reponse

1) Record line count
   (JSON format)
   The numeric value is stored key{resultdata} -> key{CONTENTS} -> key{RECORD_LENGTH}

2) Column information (Column number and column name)
   (JSON format)
   Stored as array with numeric key value starting from 0 in key{resultdata} -> key{CONTENTS} -> key{BODY} -> key{0}

Table 2-8 Response parameter list (Column information)

Column number	Column name
0	First column
1	Second column
⋮	⋮

3) Record information
   (JSON format) (One array per row (column number and column-specific data))
   Stored as array with numeric key value starting from 0 in key{resultdata} -> key{CONTENTS} -> key{BODY} -> key{(Numeric value starts from 1 to the maximum line count of the according record)}

Table 2-9 List of Response parameter (Record information)

Column number	Column data	Description
0	First data array	
1	Second data array	
⋮	⋮	

※ The Json hierarchy is same as Method:GET.

(4) EDIT(X-Command)
   Register record or update/discard/restore existing record.

・HTTP Header

Table 2-8 HTTP header parameter list

HTTP header	Value
Method	POST
X-Command	EDIT

・Parameter

1) Specification format.
   Please specify in JSON format.
   Please specify one record to one array based on the column information obtained from INFO and store the specified arrays as the element of an array encoded in JSON type

then send the array as the context of HTTP request.
For the column number 0, “execution process type”, please specify one of “Register”, “Update”, “Discard”, “Update”.

Example(1) Register

Column 0 is “execution process type”, column 1 is “Discard” , column 2 is “ID” (primary key column)
~ omitted ~
Column 10 is “Remarks, column 11 is "Last update date/time", column 12 is "Last update date/time for updating", and column 13 is "Last updated by".
In the case of adding 2 records.

▽JSON format
```
{
        "0": {
                "0": "Update",
                "1": "",
                "2": "",
                        ・・・(Omitted)・・・
                "10": "Remarks",
                "11": "",
                "12": "",
                "13": ""
        },
        "1": {
                "0": "Update",
                "1": "",
                "2": "",
                        ・・・(Omitted)・・・
                "10": "Remarks",
                "11": "",
                "12": "",
                "13": ""
        }
}
```

Example(2) Update

Column 0 is “execution process type”, column 1 is “Discard” , column 2 is “ID” (primary key column)
~ omitted ~
Column 9 is “Remarks, column 10 is "Last update date/time", column 11 is "Last update date/time for updating", and column 12 is "Last updated by".
In the case of updating the record with “ID” 10.

▽JSON format
```
{
        "0": "Update",
        "1": "",
        "2": "10",
        "9": "Remarks",
        "10": "2016/08/01 12:30:45",
        "11": "Last update date/time for updating",※
        "12": "Administrator"
}
```

※Method： Please set the "Last update date/time for updating" obtained from GET、X-Command：FILTER. This data is used to avoid old data to overwrite update new data.
"Last update date/time for updating" begins from “T_”.

Example (3)　Register (With file upload)

Column 0 is “execution process type”, column 1 is “Discard” , column 2 is “ID” (primary key column)
~ omitted ~
Column 5 is “Remarks, column 6 is "Last update date/time", column 7 is "Last update date/time for updating", and column 8 is "Last updated by".

▽JSON format
・　In the case of adding 1 record with file upload.

```
{
        "0": {
                "0": "Register",
                "3": "PV05004",
                "4": "20191226095004.yml",
                "5": "TEST"
        }
        "UPLOAD_FILE": [
                {
                        "4": "<The value of target file encoded by base64>"
                }
        ]
}
```

・　In the case of adding 2 records with file upload.

```
{
        "0": {
                "0": "Register ",
                "3": "PV05004",
                "4": "20191226095004.yml",
                "5": "TEST"
        },
        "1": {
                "0": "Register ",
                "3": "PV15004",
                "4": "20191226095004.yml",
          "5": "TEST"
     },
     "UPLOAD_FILE": [
          {
               "4": "<The value of target file encoded by base64>"
          },
          {
               "4": "<The value of target file encoded by base64>"
          }
     ]
}
```
※For UPLOAD_FILE, specify the value of target file encoded by base64 to perform file upload.
※When uploading files, please add the files in the order of element to “UPLOAD_FILE”.

・Responce

1) Execution result of each record
(JSON format)
Stored as array in key{resultdata} -> key{LIST} -> key{NORMAL} -> key{register、update、delete、error}

Table 2-9 Key parameter list

key	Value type	
name	String	The name of operation result type
ct	Numeric	Record count of each operation result

2) Operation result of each record
(JSON format)
Stored as array with numeric key value starting from 0 in key{resultdata}->key{LIST}->key{RAW}->key{Record number sent as parameter (No need to send column information), number starts from 0 as default}

Table 2-10 Key parameter list

key	Value type	
0	String	Result code (refer to the table below)
1	String	Detail code (refer to the table below)
2	String	Error message

・Response hierarchy
```
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:}
├ ・
├ ・
  ・
```

・Response

The returned response is stored in JSON format

```
{
    "status": "SUCCEED",
    "resultdata": {
        "LIST": {
            "NORMAL": {
                "register": {
                    "name": "Register",
                    "ct":
                },
                "update": {
                    "name": “Update",
                    "ct":
                },
                "delete": {
                    "name": "Discard",
                    "ct":
                },
                "revive": {
                    "name": "Restore",
                    "ct":
                },
                "error": {
                    "name": "Error",
                    "ct":
                }
            },
            "RAW": [
                [
                    ・・・Appendix: Result code/Detail code list・・・
                ],
                  ・・・(omitted)・・・
            ]
        }
    }
}
```

Appendix: Result code/Detail code list

Operation type	Result code	Detail code	Description
Register	000	201	Successfully registered.
Register	002	000	Required item not entered.
Register	002	000	Some items are duplicated with the record.
Register	002	000	There is a record that violates duplication prohibition.
Register	002	000	The length of the input value exceeds the specified number of bytes.
Register	002	000	The input value [value including NULL byte characters etc] is invalid.
Register	002	000	A value other than half-width integer was entered.
Register	002	000	Value out of range.
Register	002	000	The entered value is below the minimum value or above the maximum value.
Register	002	000	Input condition is not satisfied.
Register	002	000	A non-numeric value was entered.
Register	002	000	Tab or line feed was entered.
Register	002	000	Tab was entered.
Register	002	000	Input value is out of range.
Register	002	000	Input value is out of the range that can be processed normally by PHP function (checkdate).
Register	002	000	An unavailable value was selected.
Register	002	000	An item (primary key) that cannot be specified during registration was specified.
Register	-	-	Do not have maintenance permission.
Update	000	200	Successfully Updated.
Update	002	000	Required fields are not entered.
Update	002	000	Some items are duplicated with the record.
Update	002	000	There is a record that violates duplication prohibition.
Update	002	000	The length of the input value exceeds the specified number of bytes.
Update	002	000	The input value [value including NULL byte characters] is invalid.
Update	002	000	Value other than a half-width integer has been entered.
Update	002	000	Value out of range.
Update	002	000	The entered value is below the minimum value or above the maximum value.
Update	002	000	The input conditions do not met.
Update	002	000	A non-numeric value was entered.
Update	002	000	Tab and line feed was entered.
Update	002	000	Tab was entered.
Update	002	000	Input value was out of range.
Update	002	000	Input value was out of the range that can be processed normally by PHP function (checkdate).
Update	002	000	An unavailable value was selected.
Update	003	000	The update execution stop due to update of record from another session.
Update	003	000	An update to a discarded record was about to be performed.
Update	101	000	The column of update target was not specified.
Update	-	-	Do not have maintenance permission.
Discard	000	210	Successfully discarded.
Discard	002	000	The length of the input value exceeds the specified number of bytes.
Discard	002	000	The input value [value including NULL byte characters] is invalid.

Operation type	Result code	Detail code	Description
Discard	002	000	The entered value was below the minimum value or above the maximum value.
Discard	002	000	The input condition was not satisfied.
Discard	002	000	Tab was entered.
Discard	003	000	The discard execution stopped due to update of record from another session.
Discard	003	000	Discarding a discarded record was about to be performed.
Discard	101	000	The column of discard target was not specified.
Discard	-	-	Do not have maintenance permission
Restore	000	200	Successfully restored
Restore	002	000	Required item not entered
Restore	002	000	An item that cannot be updated was about to be updated when performing restoration.
Restore	002	000	Some items are duplicated to the record.
Restore	002	000	There is a record that violates duplication prohibition.
Restore	002	000	The length of the input value exceeds the specified number of bytes.
Restore	002	000	The input value [value including NULL byte characters] was invalid.
Restore	002	000	The entered value was below the minimum value or above the maximum value.
Restore	002	000	The input condition was not satisfied.
Restore	002	000	Tab was entered.
Restore	003	000	The restore execution stopped due to restore of record from another session.
Restore	003	000	Restoring restored record was about to be performed.
Restore	101	000	The column of restore target can't be identified.
Restore	-	-	Do not have maintenance permission.
Display	-	-	Validation error
Display	-	-	One of the following ("All records", "Exclude discarded records" and "Only discarded records") is not selected
-	000	000	Skip the operation and go on to the next record.

3 Menu export / import

3.1 RestAPI for menu export

It is possible to perform menu export with RestAPI.
The available function is same as the operation in "Export menu" menu in "ITA management console" menu group.

Table 3-1 Menu list

Menu group	Menu name	Menu ID
Management console	Export menu	2100000211

3.1.1 **Request format**

Send HTTP request with the following information.

・Path

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=MenuID

Please refer to "Table 3-2 List of parameters that can be specified for X-Command" for Menu ID.

・HTTP Header

Table 3-3 HTTP header parameter list

HTTP Header	Description
Method	POST only
Content-Type	Specify "application/json"
Authorization	When accessing the menu that needs ITA authentication, specify the value of "Login ID" and "Password"* connected with half-width colon (:) then encoded with base64.
X-Command	EXECUTE INFO These two can be chosen

Parameters that can be specified for X-Command

Table 3-4 List of parameters that can be specified for X-Command

X-Command	Description	Target menu	Menu ID
INFO	Obtain the list of menu that can be exported	Export menu	2100000211
EXECUTE	Execute menu export	Export menu	2100000211

The following is the explanation of each X-command parameter.

3.1.2 **INFO**

Output the list of the menu that can be exported.

・Parameter

No parameter to specify.

・Response

The returned response is stored in JSON format.

```
{
        "status": "SUCCEED",
        "resultdata": {
                "MENU_LIST": {
                        "Menu group ID ": {
                                "menu_group_name": "Menu group name",
                                "menu": [
                                        {
                                                "menu_id": "Menu ID ",
                                                "menu_name": "Menu name"
                                        },
                                        {
                                              ・・・(omitted)・・・
                                        }
                                ]
                        },
                        " Menu group ID ": {
                                              ・・・(omitted)・・・
                        }

                }
        }
 }
```

Table 3-5 Response item list

Item name	Remarks
Menu group ID	An array with menu group ID as key and consisted of menu.
menu_group_name	Menu group name
menu_id	Menu ID
menu_name	Menu name

3.1.3 EXECUTE

Specify target menu and execute menu export.

・Parameter

Please specify the following to “content” in JSON format.

Table 3-6 Export menu parameter

Parameter name	Value
Menu group ID	Menu ID

※ Menu group ID and Menu ID is the value in the return value of INFO.

Example) JSON description

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

・Response

The returned response is stored in JSON format. Please refer to the following for the response items.

```
{
    "status": "SUCCEED",
    "resultdata": {
        "TASK_ID": "Execution No ",
        "RESULTCODE": "Result code",
        "RESULTINFO": "Detailed information"
    }
}
```

Table 3-7 Response item list

Item name	Remarks
TASK_ID	Operation No. Users can check the execution status by searching the Operation No. in “Export/Import menu list”.
RESULTCODE	The code of execution status 000:Normal end 002:Not able to perform execution
RESULTINFO	Detailed information

3.2 RestAPI for Import menu

It is possible to perform menu import with RestAPI.
The available function is same as the operation in "Import menu" menu in "ITA management console" menu group.

Table 3-8 menu list

Menu group	Menu name	Menu ID
Management console	Import menu	2100000212

3.2.1 **Request format**

Send HTTP request with the following information.

・Path

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=MenuID

Please refer to "Table 3-9 List of parameters that can be specified for X-Command" for Menu ID.

・HTTP Header

Table 3-9 HTTP header parameter list

HTTP Header	Description
Method	POST only
Content-Type	Specify "application/json"
Authorization	When accessing the menu that needs ITA authentication, specify the value of "Login ID" and "Password"* connected with half-width colon (:) then encoded with base64.
X-Command	EXECUTE INFO These two can be chosen

Parameters that can be specified for X-Command

Table 3-9 List of parameters that can be specified for X-Command

X-Command	Description	Target menu	Menu ID
UPLOAD	Upload the exported kym file and output the list of menu that can be imported.	Import menu	2100000212
EXECUTE	Select the import target menu and execute import.	Import menu	2100000212

The following is the explanation of each X-command parameter.

3.2.2 **UPLOAD**

Upload the exported file.
Send the file encoded with base64 as parameter.

・parameter
Please specify the following in JSON format to “content.

Table 3-10 Import menu UPLOAD parameter list

Parameter name	Value
name	Target file name
base64	Specify the value of target file encoded in base64.

1）UPLOAD Json description example

```
{
   "zipfile":{
      "name":"ita_exportdata_20191224092830.kym",
      "base64":"･･･(omitted)･･･"
   }
}
```

・Response
The returned response is stored in JSON format.

```
{
      "status": ""SUCCEED",
      "resultdata": {
            "upload_id": "Upload ID ",※
            "data_portability_upload_file_name": "File name",
            "IMPORT_LIST": {
                  "Menu group ID ": {
                        "menu_group_name": "Menu group name,
                        "menu": [
                              {
                                    "menu_id": "Menu ID ",
                                    "menu_name": "Menu name"
                              }
                                ・ ・ ・(omitted)・ ・ ・

                        ],
                  " Menu group ID ": {
                                ・ ・ ・(omitted)・ ・ ・
                        ],

                        "RESULTCODE": "Result code",
                        "RESULTINFO": "Detailed information"
                  }
            }
```

```
        }
}
```

※ "upload_id" is used when executing import(EXECUTE)

Table 3-11 Response item list

Item name	Remarks
upload_id	The value given when upload succeeded. Used in EXECUTE.
data_portability_upload_file_name	File name
Menu group ID	An array with menu group ID as key and consisted of menu.
menu_group_name	Menu group name
menu_id	Menu ID
menu_name	Menu name
RESULTCODE	The code of execution status 000：Normal end 002：Not able to perform execution
RESULTINFO	Detailed information

3.2.3 EXECUTE

Perform import based on the uploaded file.
Users can specify the target menu group, menu ID, import execution mode.

・Parameter

Please specify the following items in JSON format to "content".

Table 3-12 Import menu EXECUTE parameter

Parameter name	Value	Remarks
Menu group ID	Menu ID	
importButton	Empty	Specify key when using "Import"
importButton2	Empty	Specify key when using "Import (without discarded data)"
upload_id		Add prefix "A_" to the value obtained from the return value of UPLOAD.
data_portability_upload_file_name	File name	

1）EXECUTE　Json description example

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

        "importButton": "",
        "upload_id": "A_20191217090335772040239",※
        "data_portability_upload_file_name": "ita_exportdata_20191213095733.kym"
}
```

※Use the “upload_id” obtained from UPLOAD with “A_” added in the front of the value.

・Response

The returned response is stored in JSON format. Please refer to the following for the response items.

```
{
        "status": "SUCCEED",
        "resultdata": {
                "TASK_ID": "Execution No. of menu import",
                "RESULTCODE": "Result code",
                "RESULTINFO": "Detailed information"
        }
}
```

Table 3-13 Response item list

Item name	Remarks
TASK_ID	Operation No Users can check the execution status by searching the Operation No in “Export/Import menu list”.
RESULTCODE	The code of execution status 000:Normal end 002:Not able to perform execution
RESULTINFO	Detailed information

4 Export・Import Symphony/Operation

4.1 RestAPI for Symphony/Operation export

It is possible to perform Symphony/Operation export with RestAPI.
The available function is same as the operation in "Export Symphony/Operation" menu in "ITA basic console" menu group.

Table 4-1 Menu list

Menu group	Menu name	Menu ID
Basic console	Export Symphony/Operation	2100000401

4.1.1 Request type

Send HTTP request with the following information.

・Path

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no= MenuID

Please refer to "Table 4-2 List of parameters that can be specified for X-Command" for Menu ID.

・HTTP header

Table 4-3 HTTP header parameter list

HTTP Header	Description
Method	POST only
Content-Type	Specify "application/json"
Authorization	When accessing the menu that needs ITA authentication, specify the value of "Login ID" and "Password"* connected with half-width colon (:) then encoded with base64.
X-Command	EXECUTE INFO These two can be chosen

Parameters that can be specified for X-Command

Table 4-3 List of parameters that can be specified for X-Command

X-Command	Description	Target menu	Menu ID
INFO	Obtain the list if the Symphony/Operation that can be exported.	Export Symphony/Operation	2100000401
EXECUTE	Execute export	Export Symphony/Operation	2100000401

The following is the explanation of each X-command parameter.

4.1.2 **INFO**

Output the list of the Symphony/Operation that can be exported.

・Parameter

No parameter to specify.

・Response

The returned response is stored in JSON format.

```
{
        "status": "SUCCEED",
        "resultdata": {
                "LIST": {
                        "symphony": [
                                {
                                        "SYMPHONY_CLASS_NO": "Symphony class ID",
                                        "SYMPHONY_NAME": " Symphony name",
                                 ・・・(omitted) refer to the following ① for the items・・・
                                        "LAST_UPDATE_USER": "Last update user"
                                },
                                ・・・(omitted)・・・
                        ],
                        "operation": [
                                {
                                        "OPERATION_NO_UAPK": "Operation ID",
                                        "OPERATION_NAME": " Operation name",
                                 ・・・(omitted) refer to the following ② for the items・・・
                                        "LAST_UPDATE_USER": "Last update user "
                                },
                                 ・・・(omitted)・・・
                        ]
                }
        }
}
```

① Symphony class information array

Table 4-4 Instance array list

Key	Content
SYMPHONY_CLASS_NO	Symphony class ID
SYMPHONY_NAME	Symphony name
DESCRIPTION	Description
NOTE	Remarks
DISUSE_FLAG	Discard flag (Discarded : 1)
LAST_UPDATE_TIMESTAMP	Last update date/time
LAST_UPDATE_USER	The user who performed last update

① Operation information array

Table 4-5 Movement class information list

Key	Content
OPERATION_NO_UAPK	Operation ID
OPERATION_NAME	Operation name
OPERATION_DATE	Scheduled execution date
LAST_EXECUTE_TIMESTAMP	Last execution date
NOTE	Remarks
DISUSE_FLAG	Discard flag (Discarded : 1)
LAST_UPDATE_TIMESTAMP	Last update date/time
LAST_UPDATE_USER	The user who performed last update

4.1.3 EXECUTE

Specify target menu and execute export.

・Parameter

Please specify the following to “content” in JSON format.

Table 4-6 Symphony/Operation export parameter

Parameter name	Value
symphony	Symphony class ID
operation	Operation ID

※Symphony class ID, Operation ID is obtained from return value of INFO.

Example) JSON description

```
{
        "symphony": [
                "1,2,3,4"
        ],
        "operation": [
                "1,2,3"
        ]
}
```

・Response

The returned response is stored in JSON format. Please refer to the following for the response items.

```
{
        "status": "SUCCEED",
        "resultdata": {
                "TASK_ID": "Execution No ",
                "RESULTCODE": "Result code",
                "RESULTINFO": "Detailed information"
        }
}
```

Table 4-7　Response item list

Item name	Remarks
TASK_ID	Operation No Users can check the execution status by searching the Operation No. in “Export Symphony/Operation” menu.
RESULTCODE	The code of execution status 000：Normal end 002：Not able to perform execution
RESULTINFO	Detailed information

4.2 Rest API for Import Symphony/Operation

It is possible to perform Symphony/Operation import with RestAPI.
The available function is same as the operation in "Export Symphony/Operation" menu in "ITA basic console" menu group.

Table 4-8 Menu list

Menu group	Menu name	Menu ID
Basic console	Import Symphony/Operation	2100000402

4.2.1 **Request type**

Send HTTP request with the following information.

・Path

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=MenuID

Please refer to "Table 4-10 List of parameters that can be specified for X-Command" for Menu ID.

・HTTP Header

Table 4-8 HTTP header parameter list

HTTP Header	Description
Method	POST only
Content-Type	Specify "application/json"
Authorization	When accessing the menu that needs ITA authentication, specify the value of "Login ID" and "Password"* connected with half-width colon (:) then encoded with base64.
X-Command	EXECUTE INFO These two can be chosen

Parameters that can be specified for X-Command

Table 4-10 List of parameters that can be specified for X-Command

X-Command	Description	Target menu	Menu ID
UPLOAD	Upload the exported kym2 file and output the Symphony/Operation of menu that can be imported.	Import Symphony/Operation	2100000402
EXECUTE	Select the import target Symphony/Operation and execute import.	Import Symphony/Operation	2100000402

The following is the explanation of each X-command parameter.

4.2.2 UPLOAD

Upload the exported file.
Send the file encoded with base64 as parameter.

・parameter

Please specify the following in JSON format to “content.

Table 4-9 Import Symphony/Operation UPLOAD parameter list

Parameter name	Value
name	Target file name
base64	Specify the value of target file encoded in base64.

1) UPLOAD Json description example

```
{
  "zipfile":{
    "name":"ita_exportdata_20191224092830.kym2",
    "base64":"･･･(omitted)･･･"
  }
}
```

・Response

The returned response is stored in JSON format.

```
{
	"status": ""SUCCEED",
	"resultdata": {
		"upload_id": "Upload ID ",※
		"data_portability_upload_file_name": " File name ",
		"IMPORT_LIST": [
			{
				"Operation ID": "Operation name",
				・・・(omitted)・・・
			},
			{
				"Symphony class ID": " Symphony name ",
				・・・(omitted)・・・
			}
		],
		"RESULTCODE": "000",
		"RESULTINFO": ""
	}
}
```

※ “upload_id” is used when executing import (EXECUTE)

Table 4-10 Response item list

Item name	Remarks
upload_id	The value given when upload succeeded. Used in EXECUTE.
data_portability_upload_file_name	File name
IMPORT_LIST	Operation and Symphony information stored with ID:Name array structure in the order of Operation, Symphony.
RESULTCODE	The code of execution status 000：Normal end 001：Not able to perform execution
RESULTINFO	Detailed information

4.2.3 **EXECUTE**

Perform import based on the uploaded file.
Users can specify the target Symphony, Operation, import execution mode.

・Parameter
Please specify the following items in JSON format to “content”.

Table 4-11 Import Symphony/Operation EXECUTE parameter list

Parameter name	Value	Remarks
symphony	Symphony class ID	
operation	Operation ID	
upload_id		Add prefix “A_” to the value obtained from the return value of UPLOAD.
upload_file_name	File name	

1）EXECUTE　Json description example

```
{
        "symphony": [
                "1,2,3,4"
        ],
        "operation": [
                "1,2,3"
        ],
        "upload_id": "A_20200115150157105484662З",
        "upload_file_name": "ita_exportdata_20200115150153.kym2"
}
```

※Use the “upload_id” obtained from UPLOAD with “A_” added in the front of the value.

・Response

The returned response is stored in JSON format. Please refer to the following for the response items.

```
{
        "status": "SUCCEED",
        "resultdata": {
                "TASK_ID": "Execution No. of Symphony/Operation import",
                "RESULTCODE": "Result code",
                "RESULTINFO": "Detailed information"
        }
}
```

Table 4-12　Response item list

Item name	Remarks
TASK_ID	Operation No. Users can check the execution status by searching the Operation No. in “Export/Import Symphony/Operation list”.
RESULTCODE	The code of execution status 000：Normal end 002：Not able to perform execution
RESULTINFO	Detailed information

5 Symphony

5.1 RestAPI for Symphony registration

Operating RestAPI from Symphony is possible.
The available function is same as the operation in “Symphony class editor” menu in “ITA basic console” menu group.

Table 5-1 Target menu list

Menu group	Menu name	Menu ID
Basic console	Symphony class editor	2100000306

5.1.1 **Request type**

Send HTTP request with the following information.

・Path

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no= MenuID

Please refer to “Table 5-2 List of parameters that can be specified for X-Command” for Menu ID.

・HTTP header

Table 5-3 HTTP header parameter list

HTTP Header	Description
Method	POST only
Content-Type	Specify “application/json”
Authorization	When accessing the menu that needs ITA authentication, specify the value of “Login ID” and “Password”* connected with half-width colon (:) then encoded with base64.
X-Command	INFO FILTER EDIT These three can be chosen

Parameters that can be specified for X-Command

Table 5-4 List of parameters that can be specified for X-Command

X-Command	Description	Target menu	Menu ID
INFO	Obtain the column information of Symphony class only.	Symphony class editor	2100000306
FILTER	Refer to the record that matches the parameter of Symphony class.	Symphony class editor	2100000306
EDIT	Register Symphony class.	Symphony class editor	2100000306

The following is the explanation of each X-command parameter.

5.1.2 **INFO**

Obtain the column information of Symphony class only.
※Please refer to “standard REST function” – “INFO(X-Command)” for details.

5.1.3 **FILTER**

Obtain the column information (column number and column name) and the line count, record content, and column information of all record in normal status (discarded or active) that matched the criteria specified in parameter.
※Please refer to “standard REST function” – “Filter(X-Command)” for details.

5.1.4 **EDIT**

Register, update, discard, restore Symphony class.

・HTTP header

Table 5-5 HTTP header parameter list

HTTP Header	Value
Method	POST
X-Command	EDIT

・Parameter

1) Specification format
Please refer to the following for the items that can be specified to the parameter for each execution type.
Please set the “Last update date/time for updating” obtained from X-Command:Filter for item number 7 when perform “Update”, “Discard”, or “Restore”.
This data is used to avoid old data to overwrite update new data.
"Last update date/time for updating" begins from “T_”.

Table 5-6 Symphony class parameter list

Item number	Parameter name	Remarks
0	Execution process type	Register/Update/Discard/Restore
2	Symphony class ID	Leave empty for “Register”
3	Symphony name	
4	Discription	
5	Remarks	
7	Last update date/time for updating	T_XXXXXXXXXXXXXXXXXXXX
9	Movement details	Movement information Refer to table 5-6 below for details.

Table 5-7 Movement Details list

Item number	Parameter name	Remarks
0	Orchestrator ID	The ID of Orchestrator The mapping table is in following table 5-8.
1	Movement ID	The ID of Movement Refer to “Movement list” menu
2	Pause	OFF:empty ON:checkedValue
3	Description	
4	Operation ID(Specified individually)	The ID of Operation Refer to “Input Operation list” menu

Table 5-8 Symphony class parameter Movement details

Items to specify for Parameter (Register / Update)

"9": [
    {
        "0": "Orchestrator ID",
        "1": "Movement ID",
        "2": "Pause(OFF:/ON:checkedValue)",
        "3": "Description",
        "4": "Operation ID(Specified individually)"
    },
    {
    ・・・When execute multiple Movements, add them in execution order・・・
    }
]

Table 5-9 Mapping table of Orchestrator ID

ID	Status
3	Ansible Legacy
4	Ansible Pioneer
5	Ansible Legacy Role
8	DSC
9	OpenStack

Items to specify for Parameter (Register / Update)

{
    "0": "Execution process type：<Register　or Update>",
    "2": "Symphony class ID ",
    "3": "Symphony name",
    "4": "Description",
    "5": "Remarks",
    "7": " Last update date/time for updating",
    "9": [

```
          {
               "0": "Orchestrator ID",
               "1": "Movement ID",
               "2": "Pause(OFF:/ON:checkedValue)",
               "3": "Description",
               "4": "Operation ID(Specified individually)"
               }
      ・・・When execute multiple Movements, add them in execution order・・・・
      ]
}
```

Items to specify for Parameter (Discard/restore)

```
{
     "0": " Execution process type：<Discard or Restore>",
     "2": "Symphony class ID",
     "7": "Last update date/time for updating"
}
```

※Please set the "Last update date/time for updating" obtained from Method：GET, X-Command：FILTER. This data is used to avoid old data to overwrite update new data.
"Last update date/time for updating" begins from “T_”.

Example) JSON Description example：When registering multiple execution type

```
  [
       {
            "0": "Register",
            "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": "Update",
                "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": "Discard",
                "2": 2,
                "7": "T_20191224135437197447"
        },
        {
                "0": "Restore",
                "2": 4,
                "7": "T_20191224135449793941"
        }
]
```

・Response

※Please refer to “standard REST function” – “EDIT(X-Command)” for the execution result of each record.

5.2 RestAPI for Symphony execution

Operating RestAPI from Symphony is possible.
The available function is same as the operation in “Symphony execution” and Symphony execution checking” menu in “ITA basic console” menu group.

Table 5-9 Target menu list

Menu group	Menu name	Menu ID
Basic console	Symphony class execution list	2100000309

5.2.1 **Request type**

Send HTTP request with the following information.

・Path

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=MenuID

Please refer to “Table 5-11 List of parameters that can be specified for X-Command” for Menu ID.

・HTTP Header

Table 5-10 HTTP header parameter list

HTTP Header	Description
Method	POST only
Content-Type	Specify “application/json”
Authorization	When accessing the menu that needs ITA authentication, specify the value of “Login ID” and “Password”* connected with half-width colon (:) then encoded with base64.
X-Command	EXECUTE CANCEL SCRAM RELEASE These four can be chosen

Parameters that can be specified for X-Command

Table 5-11 List of parameters that can be specified for X-Command

X-Command	Description	Target menu	Menu ID
EXECUTE	Execute Symphony	Symphony execution	2100000308
CANCEL	Cancel scheduled Symphony execution	Symphony execution checking	2100000309
SCRAM	Perform Symphony emergency stop	Symphony execution checking	2100000309
RELEASE	Release Symphony pause point	Symphony execution checking	2100000309

The following is the explanation of each X-command parameter.

5.2.2 **Response item**

The following is the explanation of the response items in each X-command parameter.

Table 5-12 Response item list

Item name	Remarks
SYMPHONY_INSTANCE_ID	Used when operating SYMPHONY instances
MOVEMENT_SEQ_NO	Used in RELEASE only
RESULTCODE	The code of execution status 000:Normal end 001:Execution unavailable 002:Scheduled cannot be cancelled 003:Cannot perform emergency stop 004:Cannot unpause
RESULTINFO	Detailed information

5.2.3 **EXECUTE**

Specify Symphony class and Operation then perform Operation execution. It is possible to specify scheduled execution date/time and skip, Operation ID to each Movement registered in Symphony class individually.

･Parameter

Please specify the following items in JSON format to "content".

Table 5-13 Operation ID individual specification parameter list

Parameter name	Value
SYMPHONY_CLASS_NO	Symphony class ID
OPERATION_ID	Operation ID
PRESERVE_DATETIME	Scheduled execution date(YYYY/MM/DD tt:mm)
OPTION	With/without skip, individual specification array of Operation ID

･Specify OPTION

In OPTION, it is possible to specify skip, Operation ID to each Movement individually in array format.

･Hierarchy of Movement element
```
├1 (Execution order of Movement)
|        ├SKIP   -   YES or NO
|        └OPERATION_ID  -  (Operation ID specified individually)
├2 (Execution order of Movement)
|        ├SKIP   -   YES or NO
|        └OPERATION_ID  -  (Operation ID specified individually)
･
･
```

1）EXECUTE　Json description example

Symphony class ID is 1, Operation ID is 1001, and scheduled date and time is 2016/01/01 00:00, In addition, skip the first executed Movement ,and specified the operation ID of the second executed Movement to 2001

▽Description in Json format

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

Figure 6.1-1 EXECUTE　Json description example

・Response

The returned response is stored in JSON format

```
{
        "status": "Successfully executed or not",
        "resultdata": {
                "SYMPHONY_INSTANCE_ID": "Execution No ",※
                "RESULTCODE": "Result code",
                "RESULTINFO": "Detailed information"
        }
}
```

※Used to operate (INFO, CANCEL, SCRAM, RELEASE) the instance after execution.

5.2.4 CANCEL

Specify the instance ID of the registered Symphony whose execution date is scheduled and cancel the schedule.

・Parameter

Please specify the following items in JSON format to “content”.

Table 5-14　Symphony execution schedule cancellation parameter list

Parameter name	Value
SYMPHONY_INSTANCE_ID	Symphony instance ID※

※The value obtained from the return value of EXECUTE.

・Response

The returned response is stored in JSON format. Please refer to the following for the returned items.

```
{
        "status": "SUCCEED",
        "resultdata": {
                " SYMPHONY_INSTANCE_ID": " Symphony instance ID during execution",
                "RESULTCODE": "Result",
                "RESULTINFO": "Detailed information"
        }
}
```

5.2.5 **SCRAM**

Specify the instance ID of the executing Symphony and perform emergency stop.

・Parameter

Please specify the following items in JSON format to “content”.

Table 5-15 Symphony execution emergency stop parameter list

Parameter name	Value
SYMPHONY_INSTANCE_ID	Symphony instance ID※

※The value obtained from the return value of EXECUTE.

・Response

The returned response is stored in JSON format. Please refer to the following for the returned items.

```
{
        "status": "SUCCEED",
        "resultdata": {
                " SYMPHONY_INSTANCE_ID": "Symphony instance ID during execution",
                "RESULTCODE": "Result",
                "RESULTINFO": "Detailed information"
        }
}
```

5.2.6 **RELEASE**

Specify Symphony instance ID and Movement order then release the points that “pause” are set.

・Parameter

Please specify the following items in JSON format to “content”.

Table 5-16 Symphony execution unpause parameter list

Parameter name	Value
SYMPHONY_INSTANCE_ID	Symphony instance ID※
MOVEMENT_SEQ_NO	The sequence number of Movement

※The value obtained from the return value of EXECUTE.

・Response

The returned response is stored in JSON format. Please refer to the following for the returned items.

```
{
        "status": "SUCCEED",
        "resultdata": {
                "SYMPHONY_INSTANCE_ID": "Symphony instance ID during execution ",
                "MOVEMENT_SEQ_NO": "The sequence number of the executed Movement in
Symphony class",
                "RESULTCODE": "Result code",
                "RESULTINFO": "Detailed information"
        }
}
```

5.3 RestAPI for Symphony execution checking

Operating RestAPI from Symphony is possible.
The available function is same as the operation in "Symphony execution" and Symphony execution checking" menu in "ITA basic console" menu group.

Table 5-17 Target menu list

Menu group	Menu name	Menu ID
Basic console	Symphony class execution checking	2100000309

5.3.1 **Request type**

Send HTTP request with the following information.

・Path

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no= MenuID

Please refer to "Table 5-19 List of parameters that can be specified for X-Command" for Menu ID.

・HTTP header

Table 5-17 HTTP header parameter list

HTTP Header	Description
Method	POST only
Content-Type	Specify "application/json"
Authorization	When accessing the menu that needs ITA authentication, specify the value of "Login ID" and "Password"* connected with half-width colon (:) then encoded with base64.
X-Command	INFO can be specified.

Parameters that can be specified for X-Command

Table 5-18 List of parameters that can be specified for X-Command

X-Command	Description	Target menu	Menu ID
INFO	Check the execution status of Symphony and return the status.	Symphony class execution checking	2100000309

The following is the explanation of each X-command parameter.

5.3.2 Response item

The following is the explanation of the response items in each X-command parameter.

Table 5-19 Response item list

Item name	Remarks
SYMPHONY_INSTANCE_ID	Used when operating SYMPHONY instances
RESULTCODE	The code of execution status 000:Normal end
RESULTINFO	Detailed information

5.3.3 INFO

Specify the instance ID of Symphony during execution and obtain the execution information.

・Parameter

Please specify the following items in JSON format to “content”.

Table 5-20 Symphony execution obtain information parameter list

Parameter name	Value
SYMPHONY_INSTANCE_ID	Symphony instance ID※

※The value obtained from the return value of EXECUTE.

・Response

The returned response is stored in JSON format.

```
{
    "status": "SUCCEED",
    "resultdata": {
        "SYMPHONY_CLASS_ID": "1",
        "SYMPHONY_INSTANCE_INFO": {
            "SYMPHONY_INSTANCE_ID": 1,
              ・・・(omitted) refer to the following ① for the items・・・
            "FOCUS_MOVEMENT": 1
        },
        "MOVEMENTS": [
            {
                "CLASS_ITEM": {
                    "ORCHESTRATOR_ID": "3",
                      ・・・(omitted) refer to the following ② for the items・・・
                    "NEXT_PENDING": "checkedValue"
                    },
                    "INS_ITEM": {
                        "STATUS": "11",
                           ・・・(omitted) refer to the following ③ for the items・・・
                        "OPERATION_NAME": null
                    }
                }
                ・・・・
            ],
            "RESULTCODE": "000",
            "RESULTINFO": ""
    }
}
```

① The Symphony instance information array stored in SYMPHONY_INSTANCE_INFO

Table 5-21 Instance array list

Key	Content
SYMPHONY_INSTANCE_ID	Symphony instance ID
I_SYMPHONY_CLASS_NO	Class ID of the instance
I_SYMPHONY_NAME	Name of the instance
I_DESCRIPTION	Description of the instance
STATUS_ID	Execution status. Refer to Table 5-25 for details
ABORT_EXECUTE_FLAG	Emergency stop flag. Not issued: 1 Issued: 2
OPERATION_NO_UAPK	Operation NO
OPERATION_NO_IDBH	Operation ID
OPERATION_NAME	Operation name
TIME_BOOK	Scheduled execution date/time
TIME_START	Start date/time
TIME_END	End date/time
MOVEMENT_LENGTH	Number of registered Movement
FOCUS_MOVEMENT	The sequence of current execution Movement

② The Movement class information stored in CLASS_ITEM

Table 5-22 Movement class information list

Key	Content
ORCHESTRATOR_ID	Orchestrator ID. Mapping table is in below table 5-26
PATTERN_ID	Movement ID
PATTERN_NAME	Movement name
THEME_COLOR	<For Web> The color of the circle icon set on Web screen
MOVEMENT_SEQ	The sequence number in Symphony class
DESCRIPTION	The description entered in Symphony class editor screen
NEXT_PENDING	If pause is set：checkedValue

③ The Movement instance information stored in INS_ITEM

Table 5-23 Movement instance information list

Key	Content
STATUS	Execution status. Refer to Table 5-25 for details
RELEASED	Pause is set: 1 Pause is released: 2
EXECUTION_NO	Movement instance ID
JUMP	<For Web> Transit target URL
ABORT_RECEPTED	Emergency stop request 1: Not received 2: received
SKIP	If skip is set:1
TIME_START	Start date/time
TIME_END	End date/time
OPERATION_ID	Individually specified Operation ID
OPERATION_NAME	Individually specified Operation Name

Table 5-24 Mapping table of status ID during Symphony instance execution

ID	Status
1	Unexecuted
2	Unexecuted (schedule)
3	Executing
4	Executing (delay)
5	Normal end
6	Emergency stop
7	Abend
8	Unexpected error
9	Schedule cancellation

Table 5-25 Mapping table of Orchestrator ID

ID	Status
3	Ansible Legacy
4	Ansible Pioneer
5	Ansible Legacy Role
8	DSC
9	OpenStack

Table 5-26 Mapping table of status ID during Movement instance execution

ID	Status
1	Unexecuted
2	Preparing
3	Executing
4	Executing(delayed)
5	Execution completed
6	Abend
7	Emergency stop
8	Holding
9	Nomal end
10	Preparation error
11	Unexpected error
12	Skip completed
13	Holding after skip
14	Skip end

6 Movement

6.1 RestAPI for Movement execution

Operating Movement from RestAPI is possible.
The available function is same as the operation in “Execution” and “Check execution status” menu in the following menu group.

Table 6-1 Execution, Check execution status menu list

Menu group	Menu name	Menu ID
Ansible-Legacy	Execution	2100020111
	Check execution status	2100020112
Ansible-Pioneer	Execution	2100020211
	Check execution status	2100020212
Ansible-LegacyRole	Execution	2100020312
	Check execution status	2100020313
DSC	Execution	2100060009
	Check execution status	2100060010
OpenStack	Execution	2100070004
	Check execution status	2100070005

6.1.1 **Request type**

Send HTTP request with the following information.

・Path

https://<HostName>:<Port>/default/menu/07_rest_api_ver1.php?no=MenuID

Please refer to “Table 6-3 List of parameters that can be specified for X-Command” for Menu ID.

・HTTP Header

Table 6-2 HTTP header parameter list

HTTP Header	Description
Method	POST only
Content-Type	Specify “application/json”
Authorization	When accessing the menu that needs ITA authentication, specify the value of “Login ID” and “Password”* connected with half-width colon (:) then encoded with base64.
X-Command	EXECUTE CANCEL SCRAM These three can be chosen.

Parameters that can be specified for X-Command

Table 6-3　List of parameters that can be specified for X-Command

X-Command	Description	Target menu	Menu ID
EXECUTE	Schedule/Execute Movement	Execution	2100020111 2100020211 2100020312 2100060009 2100070004
CANCEL	Cancel execution schedule	Check operation status	2100020112 2100020212 2100020313 2100060010 2100070005
SCRAM	Perform emergency stop	Check operation status	

The following is the explanation of each X-command parameter.

6.1.2 Response item

The following is the explanation of the response items in each X-command parameter.

Table 6-4　Response item list

Item name	Remarks
EXECUTION_NO	Used to operate with Operation No.
RESULTCODE	The code of execution status 000：Normal end 001：Not able to perform execution 002：Not able to cancel schedule 003：Not able to perform emergency stop
RESULTINFO	Detailed information

6.1.3 EXECUTE

Specify the Movement class and Operation then perform execution. Specifying scheduled execution date/time and execution mode (Dry run/Execute) is possible.

･Parameter

Please specify the following items in JSON format to “content”.

Table 6-5 Movement execution parameter list

Parameter name	Value
MOVEMENT_CLASS_ID	Movement class ID
OPERATION_ID	Operation ID
PRESERVE_DATETIME	Scheduled execution date/time (YYYY/MM/DD tt:mm)
RUN_MODE	1:Execute 2:Dry run

Example) JSON description example

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

・Response

The returned response is stored in JSON format. Please refer to the following for the response items.

```
{
    "status": "SUCCEED",
    "resultdata": {
        "EXECUTION_NO": "Operation No",
        "RESULTCODE": "Result code",
        "RESULTINFO": "Detailed information"
    }
}
```

6.1.4 CANCEL

Specify the Operation No. of the registered Operation whose execution date is scheduled and cancel the schedule.

・Parameter

Please specify the following items in JSON format to “content”.

Table 6-6 Movement execution parameter list

Parameter name	Value
EXECUTION_NO	Execution No※

※The value obtained from the return value of EXECUTE.

・Response

The returned response is stored in JSON format.

```
{
    "status": "SUCCEED",
    "resultdata": {
        "EXECUTION_NO": "Operation No",
        "RESULTCODE": "Result code",
        "RESULTINFO": "Detailed information"
    }
}
```

6.1.5 SCRAM

Specify the Operation No. of the executing Operation and perform emergency stop.

・Parameter

Please specify the following items in JSON format to “content”.

Table 6-7 Movement execution parameter list

Parameter name	Value
EXECUTION_NO	Operation No※

※The value obtained from the return value of EXECUTE.

・Response

The returned response is stored in JSON format.

```
{
        "status": "SUCCEED",
        "resultdata": {
                "EXECUTION_NO": "Operation No",
                "RESULTCODE": "Result code",
                "RESULTINFO": "Detailed information"
        }
}
```