

ITA\_User Instruction Manual

RestAPI

*－* Version 1.6*－*

Copyright © NEC Corporation 2019. All rights reserved.

Disclaimer

All the contents of this document are protected by copyright owned by NEC Corporation.

Unauthorized reproduction or copying of all or part of the contents of this document is prohibited.

The contents of this document are subject to change without prior notice in the future.

NEC Corporation is not responsible for any technical or editorial errors or omissions in this document.

NEC Corporation do not guarantee accuracy, usability, certainty of the content in this document.

Trademark

* Linux is registered trademark or trademark of Linus Torvalds, registered in the U.S. and other countries.
* Red Hat is registered trademark or trademark of Red Hat, Inc., registered in the U.S. and other countries.
* Apache, Apache Tomcat, Tomcat are registered trademarks or trademarks of Apache Software Foundation.
* Ansible is a registered trademark or trademark of the Red Hat, Inc.
* Active Directory is a registered trademark or trademark of Microsoft Corporation in the U.S. and other countries.

The names of other systems, company name and products mentioned in this document are registered trademarks or trademarks of their respective companies.

The ® mark and TM mark is not specified in this document.

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

# Table of contents

[1 Table of contents 2](#_Toc63783456)

[2 Introduction 4](#_Toc63783457)

[1 ITA system　Overview of REST API 5](#_Toc63783458)

[1.1 About REST API 5](#_Toc63783459)

[2 Standard REST function 6](#_Toc63783460)

[2.1 Format of request 6](#_Toc63783461)

[2.2 Available Method and Command 8](#_Toc63783462)

[(1) GET (Method) 8](#_Toc63783463)

[(2) INFO(X-Command) 10](#_Toc63783464)

[(3) FILTER(X-Command) 10](#_Toc63783465)

[(4) EDIT(X-Command) 13](#_Toc63783466)

[3 Menu export / import 20](#_Toc63783467)

[3.1 RestAPI for menu export 20](#_Toc63783468)

[3.1.1 Request format 20](#_Toc63783469)

[3.1.2 INFO 21](#_Toc63783470)

[3.1.3 EXECUTE 22](#_Toc63783471)

[3.2 RestAPI for Import menu 24](#_Toc63783472)

[3.2.1 Request format 24](#_Toc63783473)

[3.2.2 UPLOAD 25](#_Toc63783474)

[3.2.3 EXECUTE 26](#_Toc63783475)

[4 Export・Import Symphony/Operation 28](#_Toc63783476)

[4.1 RestAPI for Symphony/Operation export 28](#_Toc63783477)

[4.1.1 Request type 28](#_Toc63783478)

[4.1.2 INFO 29](#_Toc63783479)

[4.1.3 EXECUTE 30](#_Toc63783480)

[4.2 Rest API for Import Symphony/Operation 32](#_Toc63783481)

[4.2.1 Request type 32](#_Toc63783482)

[4.2.2 UPLOAD 33](#_Toc63783483)

[4.2.3 EXECUTE 34](#_Toc63783484)

[5 Symphony 36](#_Toc63783485)

[5.1 RestAPI for Symphony registration 36](#_Toc63783486)

[5.1.1 Request type 36](#_Toc63783487)

[5.1.2 INFO 37](#_Toc63783488)

[5.1.3 FILTER 37](#_Toc63783489)

[5.1.4 EDIT 37](#_Toc63783490)

[5.2 RestAPI for Symphony execution 41](#_Toc63783491)

[5.2.1 Request type 41](#_Toc63783492)

[5.2.2 Response item 42](#_Toc63783493)

[5.2.3 EXECUTE 42](#_Toc63783494)

[5.2.4 CANCEL 43](#_Toc63783495)

[5.2.5 SCRAM 44](#_Toc63783496)

[5.2.6 RELEASE 44](#_Toc63783497)

[5.3 RestAPI for Symphony execution checking 45](#_Toc63783498)

[5.3.1 Request type 45](#_Toc63783499)

[5.3.2 Response item 46](#_Toc63783500)

[5.3.3 INFO 46](#_Toc63783501)

[6 Conductor 50](#_Toc63783502)

[6.1 RestAPI for executing Conductor operations. 50](#_Toc63783503)

[6.1.1 Request format. 50](#_Toc63783504)

[6.1.2 Response Items 51](#_Toc63783505)

[6.1.3 EXECUTE 51](#_Toc63783506)

[6.1.4 CANCEL 52](#_Toc63783507)

[6.1.5 SCRAM 53](#_Toc63783508)

[6.1.6 RELEASE 53](#_Toc63783509)

[6.2 RestAPI for Conductor confirmation 54](#_Toc63783510)

[6.2.7 Request format 54](#_Toc63783511)

[6.2.8 Response items 54](#_Toc63783512)

[6.2.9 INFO 55](#_Toc63783513)

[7 Movement 58](#_Toc63783514)

[7.1 RestAPI for Movement execution 58](#_Toc63783515)

[7.1.1 Request type 58](#_Toc63783516)

[7.1.2 Response item 59](#_Toc63783517)

[7.1.3 EXECUTE 59](#_Toc63783518)

[7.1.4 CANCEL 60](#_Toc63783519)

[7.1.5 SCRAM 60](#_Toc63783520)

[8 Appendix 62](#_Toc63783521)

[8.1 Troubleshooting 62](#_Toc63783522)

# Introduction

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

# ITA system　Overview of REST API

This chapter explains the standard REST API for operating ITA.

## 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 |
| Symphony | Symphony class editor | 2100000306 | 5 Symphony |
| Symphony execution | 2100000308 |
| Symphony execution checking | 2100000309 |
| Export/Import | Export menu | 2100000211 | 3 Menu export / import |
| Import menu | 2100000212 |
| Export Symphony/Operation | 2100000401 | 4 Export・Import Symphony/Operation |
| Import Symphony/Operation | 2100000402 |
| Ansible-Legacy | Execution | 2100020111 | 7 Movement |
| Check operation status | 2100020112 |
| Ansible-Pioneer | Execution | 2100020211 |
| Check operation status | 2100020212 |
| Ansible-LegacyRole | Execution | 2100020312 |
| Check operation status | 2100020313 |
| OpenStack | Execution | 2100070004 |
| Check operation status | 2100070005 |
| Terraform | Execution | 2100080009 |
| Check operation status | 2100080010 |

# Standard REST function

Using REST API from external programs to operate the resources managed in ITA is possible.

The following shows the calling convention.

## 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

## 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

#### 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}

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{BODY} -> key{0}

Table 2-3 Response parameter list (Column information)

|  |  |
| --- | --- |
| Column number | Column name |
| 0 | First column |
| 1 | Second column |
| … | … |

1. 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"  ]  }  }  }  } |

#### 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

#### 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.

1. 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}

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{BODY} -> key{0}

Table 2-8 Response parameter list (Column information)

|  |  |
| --- | --- |
| Column number | Column name |
| 0 | First column |
| 1 | Second column |
| … | … |

1. 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.

#### EDIT(X-Command)

Register record or update/discard/restore existing record.

・HTTP Header

Table 2-10 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”.   
※If you want to change the size limit of file uploads, you need to change the PHP settings. Refer to section “8.1 Troubleshooting” for information on what items to change.   
Refer to “Reference” configuration settings when installing” for more information on default values (different document).

・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 2-11 Key parameter list

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

1. 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 2-12 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. |

# Menu export / import

## 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 “Export/Import” menu group.

Table 3-1 Menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Export/Import | Export menu | 2100000211 |

### 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 3-3　List of parameters that can be specified for X-Command” for Menu ID.

・HTTP Header

Table 3-2 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  INFO  These two can be chosen |

Parameters that can be specified for X-Command

**Table 3-3　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.

### 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 3-4　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 |

### EXECUTE

Specify target menu and execute menu export.

・Parameter

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

**Table 3-5 Export menu parameter**

|  |  |
| --- | --- |
| Parameter name | Value |
| Menu group ID | Menu ID |
| Dp\_mode | Mode 1.Override 2.Add |
| Abolished\_type | Abolition data 1.Normal 2.Without disuse data |

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

|  |
| --- |
| **Example) JSON description** |
| {  "2100000002": [  2100000202,  ・・・(omitted)・・・  2100000222  ],  "2100000003": [  ・・・(omitted)・・・  ],  “dp\_mode”:”1”,  “abolished\_type”:”1”  } |
|
|

・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 3-6　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 |

## 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 “Export/Import” menu group.

Table 3-7 menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Export/Import | Import menu | 2100000212 |

### 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 3-8 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 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.

### 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",  “dp\_mode”:”1”,  “abolished\_type”:”1”,  "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 |
| Dp\_mode | Mode 1:Override 2:Add |
| Abolished\_type | Abolition information 1:Normal 2:Without disuse data |
| 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 |

### 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 |  |
| 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 |

# Export・Import Symphony/Operation

## 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 “Export/Import” menu group.

Table 4-1 Menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Export/Import | Export Symphony/Operation | 2100000401 |

### 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 4-3　List of parameters that can be specified for X-Command” for Menu ID.

・HTTP header

Table 4-2 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  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.

### 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 |

### 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 |

## 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 “Export/Import” menu group.

Table 4-8 Menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Export/Import | Import Symphony/Operation | 2100000402 |

### 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 4-9　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 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.

### UPLOAD

Upload the exported file.

Send the file encoded with base64 as parameter.

・parameter

Please specify the following in JSON format to “content.

**Table 4-11　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 4-12　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 |

### EXECUTE

Perform import based on the uploaded file.

Users can specify the target Symphony class ID and Operation ID.

・Parameter

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

**Table 4-13 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\_202001151501571054846623",  "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 4-14　 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 |

# Symphony

## RestAPI for Symphony registration

Operating RestAPI from Symphony is possible.

The available function is same as the operation in “Symphony class editor” menu in “Symphony” menu group.

Table 5-1 Target menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Symphony | Symphony class editor | 2100000306 |

### 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 5-3　List of parameters that can be specified for X-Command” for Menu ID.

・HTTP header

Table 5-2 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 | INFO  FILTER  EDIT  These three can be chosen |

Parameters that can be specified for X-Command

**Table 5-3**　 **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.

### INFO

Obtain the column information of Symphony class only.

※Please refer to “standard REST function” – “INFO(X-Command)” for details.

### 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.

### EDIT

Register, update, discard, restore Symphony class.

・HTTP header

Table 5-4 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 5-5 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 5-6 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 5-7 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 5-8　Mapping table of Orchestrator ID

|  |  |
| --- | --- |
| ID | Status |
| 3 | Ansible Legacy |
| 4 | Ansible Pioneer |
| 5 | Ansible Legacy Role |
| 9 | OpenStack |
| 10 | Terraform |

|  |
| --- |
| **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.

## 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 “Symphony” menu group.

Table 5-9 Target menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Symphony | Symphony class execution list | 2100000309 |

### 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.

### 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 |

### 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.

### 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"  }  } |
|
|

### 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"  }  } |
|
|

### 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"  }  } |
|
|

## 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 “Symphony” menu group.

Table 5-17 Target menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Symphony | Symphony class execution checking | 2100000309 |

### 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 5-18 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 5-19　 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.

### Response item

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

Table 5-20　 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 |

### 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 5-21　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 5-22　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 |

1. The Movement class information stored in CLASS\_ITEM

Table 5-23　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 |

1. The Movement instance information stored in INS\_ITEM

Table 5-24　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 5-25　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 5-26　Mapping table of Orchestrator ID

|  |  |
| --- | --- |
| ID | Status |
| 3 | Ansible Legacy |
| 4 | Ansible Pioneer |
| 5 | Ansible Legacy Role |
| 9 | OpenStack |
| 10 | Terraform |

Table 5-27 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 |

# Conductor

## RestAPI for executing Conductor operations.

It is possible to control Conductor in RestAPI.

The functions possible to control are the ones found in the (Conductor Menu) group-> (Conductor Execution) menu -> (Conductor Confirmation) menu.

Table 6-1. Target menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Conductor | Conductor Execution | 2100180004 |
| Conductor Execution check | 2100180005 |

### Request format.

Execute HTTP Request with the information below

・Path

https://<HostName>:<Port>/default/menu/07\_rest\_api\_ver1.php?no=MenuID

Please refer to “Table 6-3. Parameters that can be specified to X-command list” for the Menu ID

・HTTP Header

Table 6‑1 HTTP Header parameter list.

|  |  |
| --- | --- |
| HTTPヘッダ | 説明 |
| Method | POST only |
| Content-Type | Specifies “application/json” |
| Authorization | In order to access ITA Menus that requires authentication, concatenate Login ID and Password with half-width colon ( : ) and Base64econdoded value |
| X-Command | Choose between  EXECUTE  CANCEL  SCRAM  RELEASE |

Table 6‑2　Parameters that can be specified to X-command list

|  |  |  |  |
| --- | --- | --- | --- |
| X-Command | Description | Screen | Menu ID |
| EXECUTE | Executes Conductor operation | Conductor Execution | 2100180004 |
| CANCEL | Deletes Conductor reservation. | Conductor confirmation | 2100180005 |
| SCRAM | Initiates emergency stop for Conductor. | Conductor confirmation | 2100180005 |
| RELEASE | Resumes paused Conductor. | Conductor confirmation | 2100180005 |

In the following section, each X-command parameter will be explained.

### Response Items

In the following section, the response items for executing X-commands will be explained.

Table 6‑3　Response Item list

|  |  |
| --- | --- |
| Item | Remarks |
| CONDUCTOR\_INSTANCE\_ID | Used to operate on Conductor instances.. |
| NODE\_INSTANCE\_NO | Only used when RELEASE |
| RESULTCODE | Result codes for Command execution  000：Normal end  001: Cannot be executed  002：Cannot delete reservation  003：Cannot initiate Emergency stop  004：Cannot resume from pause |
| RESULTINFO | Detailed information |

### EXECUTE

Select desired Conductor class and Operation and execute. It is possible to specify the reservation time/date, Skip and Operation ID for each movement registered in Conductor class.

・Parameter

　Specify the following as” content” in JSON format

Table 6‑4　Individually specified Operation ID parameter list.

|  |  |
| --- | --- |
| Parameter | Values |
| CONDUCTOR\_CLASS\_NO | Conductor class ID |
| OPERATION\_ID | Operation ID |
| PRESERVE\_DATETIME | Reservation Data/Time（YYYY/MM/DD tt:mm） |

|  |
| --- |
| 1）EXECUTE　Json Description example |
| When the Conductor class ID is 1, Operation ID is 1001 and the reservation date/time is 2016/01/01 00:00  ▽Written in JSON Format  {  "CONDUCTOR\_CLASS\_NO": 1,  "OPERATION\_ID": 1001,  "PRESERVE\_DATETIME": "2016/01/01 00:00",  } |

**Figure 6.1-1 EXECUTE　Json Description example**

　・Response

　　The return response is stored in JSON format.

|  |
| --- |
| {  "status": "Execution success/failure",  "resultdata": {  "CONDUCTOR\_INSTANCE\_ID": "Execution No ",※  "RESULTCODE": "Result Code",  "RESULTINFO": "Detailed Information"  }  } |
|
|

※Use when controlling these instances after execution (INFO, CANCEL, SCRAM, RELEASE)

### CANCEL

Cancels the specified Conductor ID that has a registered reservation date/time.

・Parameter

　　Specify the following as “content” in JSON format.

Table 6‑5　 Conductor Execution Cancel Parameter table.

|  |  |
| --- | --- |
| Parameter | Value |
| CONDUCTOR\_INSTANCE\_ID | Conductor Instance ID※ |

※Obtained by EXECUTE return value.

・Response

　　Return responses are stored in JSON format. See below for information about items

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  " CONDUCTOR\_INSTANCE\_ID": " Conductor ID when executed ",  "RESULTCODE": "Result code",  "RESULTINFO": "Detailed Information"  }  } |
|
|

### SCRAM

Select Conductor and Node instance IDs and un-pause any points that are set to pause.

・Parameter

　　Specify the following as “content” in JSON format.

Table 6‑6 Conductor Process Pause release Parameter table

|  |  |
| --- | --- |
| Parameter | Value |
| CONDUCTOR\_INSTANCE\_ID | Conductor instance ID※ |

※Obtained by EXECUTE return value.

・Response

　　Return responses are stored in JSON format. See below for information about items.

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  " CONDUCTOR\_INSTANCE\_ID": " Conductor ID when executed ",  "RESULTCODE": "Result code",  "RESULTINFO": "Detailed information"  }  } |
|
|

### RELEASE

Select Conductor and Node instance ID and un-pause any points that are set to pause.

・Parameter

　　Specify the following as “content” in JSON format.

Table 6‑7 Conductor process pause release parameter table

|  |  |
| --- | --- |
| Parameter | Value |
| CONDUCTOR\_INSTANCE\_ID | Conductor instance ID※1 |
| NODE\_INSTANCE\_ID | Node instance ID※2 |

※1 Obtained by EXECUTE Return value

※2 Refer to “6.2 RestAPI for Conductor confirmation” for obtaining Node Instance ID.

・Response

　　The returned response is stored in JSON format. See below for information regarding the items.

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "CONDUCTOR\_INSTANCE\_ID": "Conductor ID when executed",  " NODE\_INSTANCE\_NO": "Node instance ID of (Conductor pause)",  "RESULTCODE": "Result Code",  "RESULTINFO": "Detailed information"  }  } |
|
|

## RestAPI for Conductor confirmation

Conductor can be controlled from RestAPI。

The functions possible to control are the functions in (Conductor) Menu group-> (Conductor Execution) menu-> (Conductor Confirmation) menu.

Table 6‑8　Target menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Conductor | Conductor confirmation | 2100180005 |

### Request format

Execute HTTP Request with the following information.

・Path

https://<HostName>:<Port>/default/menu/07\_rest\_api\_ver1.php?no=MenuID

For menu ID, please refer to “Table 6-9 List of parameters that can be specified for X-Command

・HTTP header

Table 6‑10 HTTP header parameter list

|  |  |
| --- | --- |
| HTTP header | Description |
| Method | POST only |
| Content-Type | Specify “application/json”. |
| Authorization | In order to access ITA menus that requires authentication, concatenate Login ID and Password with half-width colon ( : ) and Base64encoded value. |
| X-Command | Can choose the following: INFO |

Parameters that can be specified to to X-command.

Table 6‑11　Parameters that can be specified for X-command list

|  |  |  |  |
| --- | --- | --- | --- |
| X-Command | Description | Menu | Menu ID |
| INFO | Checks the status of the Conductor and returns it. | Conductor Confirmation | 2100180005 |

In the following section, the response items for each running X-command will be explained.

### Response items

In the following section, the response items for each running X-command will be explained.

Table 6‑12　Response item list

|  |  |
| --- | --- |
| Item | Remarks |
| CONDUCTOR\_INSTANCE\_ID | Use to operate on CONDUCTOR instances. |
| RESULTCODE | Command execution success/failure  000：Normal end |
| RESULTINFO | Detailed information |

### INFO

Specify the instance ID when executing Conductor to obtain runtime information.

・Parameter

　　Specify the following as “content” in JSON format.

Table 6‑13　Conductor Execution information acquisition parameter table

|  |  |
| --- | --- |
| Parameter | Value |
| CONDUCTOR\_INSTANCE\_ID | Conductor Instance ID※ |

※Obtained by EXECUTE return value.

・Response

　　The return response is stored in JSON format.

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "CONDUCTOR\_INSTANCE\_INFO": {  "CONDUCTOR\_INSTANCE\_ID": 1,  ・・・（abbr.）See (1) below for information about items・・・  },  "NODE\_INFO": [  "node-1": {  "NODE\_NAME": "11",  ・・・（abbr.）See (2) below for information about items・・・  }  }  　　　　　　・・・・  ],  "RESULTCODE": "000",  "RESULTINFO": ""  }  } |
|
|

* + - 1. Conductor instance information array stored in CONDUCTOR\_INSTANCE\_INFO

Table 6‑14　Instance array table

|  |  |
| --- | --- |
| Key | Content |
| CONDUCTOR\_INSTANCE\_ID | Conductor instance ID |
| CONDUCTOR\_CLASS\_NO | ID of the original class of this instance |
| STATUS\_ID | Detailed execution status. More in the table below　“Table 6‑16” |
| EXECUTION\_USER | Execution user |
| ABORT\_EXECUTE\_FLAG | Emergency stop flag.　Not issued：1 Issued：2 |
| OPERATION\_NO\_IDBH | Registered operation ID |
| OPERATION\_NAME | Registered operation name |
| TIME\_BOOK | Reserved time/date |
| TIME\_START | Start time/date |
| TIME\_END | End time/date |

* + - 1. Node instance information that are going to be stored in NODE\_INFO

Table 6‑15　Node instance information table

|  |  |
| --- | --- |
| Key | Content |
| NODE\_NAME | Node name |
| NODE\_INSTANCE\_NO | Node instance No |
| NODE\_TYPE\_ID | Node type ID. More in the table below Table 6‑18” |
| STATUS | Node status. More in the table below “Table 6‑19” |
| SKIP | Skip is set　：2 |
| TIME\_START | Start time/date |
| TIME\_END | End time/date |
| OPERATION\_ID | Individually specified Operation ID |
| OPERATION\_NAME | Individually specified Operation Name |

1. Target ID Table

Table 6‑16　Status ID of Conductor instance when running.

|  |  |
| --- | --- |
| ID | Status |
| 1 | Not executed |
| 2 | Not executed (reserved) |
| 3 | Running |
| 4 | Running (Extended) |
| 5 | Normal end |
| 6 | Emergency stop |
| 7 | Abnormal end |
| 8 | Unexpected error |
| 9 | Delete reservation |

Table 6‑17 Orchestra ID table

|  |  |
| --- | --- |
| ID | Status |
| 3 | Ansible Legacy |
| 4 | Ansible Pioneer |
| 5 | Ansible Legacy Role |
| 10 | Terraform |

Table 6‑18 Node type ID table

|  |  |
| --- | --- |
| ID | Status |
| 1 | Conductor start |
| 2 | Conductor end |
| 3 | Movement |
| 4 | Conductor call |
| 5 | Parallel branch |
| 6 | Conditional branch |
| 7 | Parallel merge |
| 8 | Conductor pause |
| 10 | Symphony call |

Table 6‑19 Node instance runtime status.

|  |  |
| --- | --- |
| ID | Status |
| 1 | Not executed |
| 2 | Preparing |
| 3 | Running |
| 4 | Running (extended) |
| 5 | Finished |
| 6 | Abnormal end |
| 7 | Emergency stop |
| 8 | On hold |
| 9 | Normal end |
| 10 | Preparation error |
| 11 | Unexpected error |
| 12 | Skip complete |
| 13 | On hold after skip |
| 14 | Skip finished |

# Movement

## 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 7-1 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 |
| Terraform | Execution | 2100080009 |
| Check execution status | 2100080010 |

### 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 7-2 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 7-3　 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  2100080009 |
| CANCEL | Cancel execution schedule | Check operation status | 2100020112  2100020212  2100020313  2100080010 |
| SCRAM | Perform emergency stop | Check operation status |

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

### Response item

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

Table 7-4　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 |

### 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 7-5 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"  }  } |
|
|

### 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 7-6 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"  }  } |
|
|

### 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 7-7 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"  }  } |
|
|

# Appendix

## Troubleshooting

| **No** | **Content** |
| --- | --- |
| Q-1 | Uploading files using RestAPI takes too long.  An error occurs during the registration process when using RestAPI.  The display and operation web screen becomes slow when uploading files with RestAPI. |
| A-1 | It is possible that your PHP memory settings are set too low. Please check the values of the following parameters in the PHP configuration file (php.ini) and increase the set maximum value.  ・memory\_limit Memory available for PHP allocation  ・post\_max\_size Maximum size allowed for post data. |