

ITA\_User Instruction Manual

RestAPI

*－* Version 1.9*－*

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

[Table of contents 2](#_Toc96600431)

[Introduction 5](#_Toc96600432)

[1 ITA system – RestAPI overview 6](#_Toc96600433)

[1.1 About REST API 6](#_Toc96600434)

[1.1.1 Request format 6](#_Toc96600435)

[1.1.2 INFO 7](#_Toc96600436)

[1.1.3 FILTER・FILTER\_DATAONLY 7](#_Toc96600437)

[1.1.4 DOWNLOAD 7](#_Toc96600438)

[(1) UPLOAD\_SPREADSHEET(X-Command) 11](#_Toc96600439)

[2 Standard REST function 15](#_Toc96600440)

[2.1 Format of request 15](#_Toc96600441)

[2.2 Available Methods and Commands 17](#_Toc96600442)

[(2) GET (Method) 17](#_Toc96600443)

[(3) INFO(X-Command) 19](#_Toc96600444)

[(4) FILTER(X-Command) 19](#_Toc96600445)

[(5) EDIT(X-Command) 23](#_Toc96600446)

[(1) LIST\_OPTIONS(X-Command) 29](#_Toc96600447)

[(2) UPLOAD\_SPREADSHEET(X-Command) 31](#_Toc96600448)

[3 Menu export / import 34](#_Toc96600449)

[3.1 RestAPI for menu export 34](#_Toc96600450)

[3.1.1 Request format 34](#_Toc96600451)

[3.1.2 INFO 35](#_Toc96600452)

[3.1.3 EXECUTE 36](#_Toc96600453)

[3.2 RestAPI for Import menu 38](#_Toc96600454)

[3.2.1 Request format 38](#_Toc96600455)

[3.2.2 UPLOAD 38](#_Toc96600456)

[3.2.3 EXECUTE 40](#_Toc96600457)

[4 Export/Import in bulks using Excel 42](#_Toc96600458)

[4.1 RestAPI for exporting in bulks using Excel 42](#_Toc96600459)

[4.1.1 Request format 42](#_Toc96600460)

[4.1.2 INFO 43](#_Toc96600461)

[4.1.3 EXECUTE 44](#_Toc96600462)

[4.2 RestAPI for importing in bulks using Excel 46](#_Toc96600463)

[4.2.4 Request format 46](#_Toc96600464)

[4.2.5 UPLOAD 47](#_Toc96600465)

[4.2.6 EXECUTE 49](#_Toc96600466)

[5 Using Symphony 51](#_Toc96600467)

[5.1 RestAPI for registering operations for Symphony. 51](#_Toc96600468)

[5.1.1 Request type 51](#_Toc96600469)

[5.1.2 INFO 52](#_Toc96600470)

[5.1.3 FILTER 52](#_Toc96600471)

[5.1.4 EDIT 52](#_Toc96600472)

[5.2 RestAPI for Symphony execution 55](#_Toc96600473)

[5.2.1 Request type 56](#_Toc96600474)

[5.2.2 Response item 57](#_Toc96600475)

[5.2.3 EXECUTE 57](#_Toc96600476)

[5.2.4 CANCEL 58](#_Toc96600477)

[5.2.5 SCRAM 59](#_Toc96600478)

[5.2.6 RELEASE 59](#_Toc96600479)

[5.3 RestAPI for Symphony execution checking 60](#_Toc96600480)

[5.3.1 Request type 60](#_Toc96600481)

[5.3.2 Response item 61](#_Toc96600482)

[5.3.3 INFO 61](#_Toc96600483)

[5.4 RestAPI for Symphony list 64](#_Toc96600484)

[5.4.4 Request format 64](#_Toc96600485)

[5.4.5 INFO 65](#_Toc96600486)

[5.4.6 FILTER・FILTER\_DATAONLY 65](#_Toc96600487)

[5.4.7 DOWNLOAD 65](#_Toc96600488)

[6 Conductor 67](#_Toc96600489)

[6.1 RestAPI for registering Conductor operations 67](#_Toc96600490)

[6.1.1 Request format. 67](#_Toc96600491)

[6.1.2 INFO 68](#_Toc96600492)

[6.1.3 FILTER・FILTER\_DATAONLY 71](#_Toc96600493)

[6.1.4 EDIT 76](#_Toc96600494)

[6.2 RestAPI for executing Conductor operations 80](#_Toc96600495)

[6.2.1 Request format. 81](#_Toc96600496)

[6.2.2 Response Items 81](#_Toc96600497)

[6.2.3 EXECUTE 82](#_Toc96600498)

[6.2.4 CANCEL 83](#_Toc96600499)

[6.2.5 SCRAM 84](#_Toc96600500)

[6.2.6 RELEASE 84](#_Toc96600501)

[6.3 RestAPI for Conductor confirmation 85](#_Toc96600502)

[6.3.1 Request format 85](#_Toc96600503)

[6.3.2 Response items 86](#_Toc96600504)

[6.3.3 INFO 86](#_Toc96600505)

[6.4 RestAPI for Conductor list 89](#_Toc96600506)

[6.4.4 Request format 89](#_Toc96600507)

[6.4.5 INFO 89](#_Toc96600508)

[6.4.6 FILTER・FILTER\_DATAONLY 89](#_Toc96600509)

[6.4.7 DOWNLOAD 90](#_Toc96600510)

[7 Movement 92](#_Toc96600511)

[7.1 RestAPI for Movement execution 92](#_Toc96600512)

[7.1.1 Request type 92](#_Toc96600513)

[7.1.2 Response item 93](#_Toc96600514)

[7.1.3 EXECUTE 93](#_Toc96600515)

[7.1.4 CANCEL 94](#_Toc96600516)

[7.1.5 SCRAM 94](#_Toc96600517)

[8 Version confirmation 96](#_Toc96600518)

[8.1 RestAPI for checking version 96](#_Toc96600519)

[8.1.1 Request format 96](#_Toc96600520)

[8.1.2 INFO 96](#_Toc96600521)

[9 Compare function 98](#_Toc96600522)

[9.1 RestAPI for the Compare function. 98](#_Toc96600523)

[9.1.1 Request format 98](#_Toc96600524)

[9.1.2 COMPARE 99](#_Toc96600525)

[10 Appendix 101](#_Toc96600526)

[10.1 Troubleshooting 101](#_Toc96600527)

# Introduction

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

# ITA system – RestAPI overview

This chapter explains the standard REST API for operating ITA.

## About REST API

ITA provides REST API for various operations and 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 Using Symphony |
| Symphony execution | 2100000308 |
| Symphony execution checking | 2100000309 |
| Conductor | Conductor execution | 2100180004 | 5.4 RestAPI for Symphony list Request format Send HTTP request with the following information.  ・Path  https://<HostName>:<Port>/default/menu/07\_rest\_api\_ver1.php?no=Menu ID  See “Table 5-28 List of parameters that can be specified for X-Command” for the different Menu IDs  ・HTTP header  Table 5‑29 HTTP header parameter list   |  |  | | --- | --- | | HTTP header | Description | | Content-Type | Specifies “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 | The following can be selected: INFO  FILTER  FILTER\_DATAONLY  DOWNLOAD |   Parameters that can be specified for X-Command  Table 5‑30　Parameters that can be specified for X-Command   |  |  |  |  | | --- | --- | --- | --- | | X-Command | Description | Target screen | Menu ID | | INFO | Acquires Symphony class information | Symphony list | 2100000310 | | FILTER  FILTER\_DATAONLY | References records that matches Symphony class parameters | Symphony list | 2100000310 | | DOWNLOAD | Acquires result data and input data from the Symphony list. | Symphony list | 2100000310 |   The following explains the different X-command parameters. INFO Acquires Symphony class column information.  ※See ”Standard REST function”-“INFO(X-Command)” for more information FILTER・FILTER\_DATAONLY Acquires either the column information of the records that matches the set conditions or column information, record contents and line numbers for all the records with normal status (Abolished or restoring).  If FILTER\_DATAONLY is specified, the column information will be shortened.  ※See ”Standard REST function”-“INFO(X-Command)” for more information DOWNLOAD Acquires a BASE64-encoded zip file containing a set of input data and a set of result data for records that match the conditions specified by the parameters.  **Table 5‑20** **Symphony class parameter list**   |  |  | | --- | --- | | **Item key name** | **Parameter name** | | SYMPHONY\_INSTANCE\_NO | Comma-separated Synphony instance number desired record |  |  | | --- | | **Example）JSON description：Symphony Instance number 1 and 2** | | {  "SYMPHONY\_INSTANCE\_NO": [  　 "1",  "2"  ]  } | | | | | |   ・Response  1) Record line number  （JSON format）  Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{RECORD\_LENGTH} as numeric value  2) Record information  （JSON format）（1 array per line1（Column key and Column data））  Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{BODY} -> Key{0} as an array whose keys are numbers starting from 0.  **Table 5‑21** **Response parameter list**   |  |  | | --- | --- | | Column key name | Column name | | SYMPHONY\_INSTANCE\_NO | Symphony instance No | | INPUT\_DATA | Input data file name | | RESULT\_DATA | Result data file name |   3) File information  （JSON format）（1 array per line1（Column key and Column data）  Key{resultdata} -> Key{CONTENTS} -> Key{ DOWNLOAD\_FILE} -> Key{0} as an array whose keys are numbers starting from 0.  **Table 5‑21** **Response parameter list**   |  |  | | --- | --- | | Column Key name | Column name | | Input data file name | BASE64-encoded value of the downloaded file | | Result data file name | BASE64-encoded value of the downloaded file |   ・Response  The returned response is stored in JSON format   |  | | --- | | {  "status": "SUCCEED",  "resultdata": {  "CONTENTS": {  “RECORD\_LENGTH”:Acquired number  "BODY": {  "0":{  "SYMPHONY\_INSTANCE\_NO": Synphony Instance No  "INPUT\_DATA ":Input data file name  "RESULT\_DATA":Result data file name  },  ・・・Repeat for number of acquisitions・・・  }  "DOWNLOAD\_FILE:{  "0"{  "Input data file name": BASE64-encoded value of the downloaded file  "Result data file name": BASE64-encoded value of the downloaded file  }  ・・・Repeat for number of acquisitions・・・  }  }  }  } |   Conductor |
| Conductor execution checking | 2100180005 |
| Export/Import | Export menu | 2100000211 | 0  LIST\_OPTIONS(X-Command)  Acquires a list of values that can be selected from the pulldown selections from the different menus.  ※The selection value of Access permission/Access permission roles cannot be acquired.  Table 2‑12 HTTP header parameter list   |  |  | | --- | --- | | HTTP header | value | | X-Command | LIST\_OPTIONS |   ・Response  1)Column information（number and name）  （JSON format）  Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{BODY} -> Key{0}As a key starting with the numeric value “0”  Table 2‑13 Response parameter list (Column information)   |  |  | | --- | --- | | Column number | Column name | | 0 | Column 1 | | 1 | Column 2 | | … | … |   2)Record information  （JSON format）  Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{BODY} -> Key{1} as an array  Only data from pulldown items will be acquired. Other items will be blank.  Table 2‑14 Response parameter list (Record information)   |  |  |  | | --- | --- | --- | | Column number | Column data | Description | | 0 | Column 1 data |  | | 1 | Column 2 data |  | | … | … |  |   ・Response  The returned response is stored in JSON format.   |  | | --- | | {  "status": "SUCCEED",  "resultdata": {  "CONTENTS": {  "BODY": {  0: {  0:"Execution process type"  1:・・・  2:・・・  },  1: {  0:{  0:"Register”  1:"Update”  2:"Abolish”  3:"Restore”  }  1:{Table if item is pulldown、Blank if otherwise}  ・・・  }  }  }  } |   DOWNLOAD\_SPREADSHEET(X-Command)  Download menus as excel files with the filter function active. If the menu exceeds the maximum excel line number, it will be downloaded as an SCSV file.  The parameter specification is the same as FILTER  Table 2‑15 HTTP header parameter   |  |  | | --- | --- | | HTTP Header | Value | | X-Command | DOWNLOAD\_SPREADSHEET |   ・Response  1) Record information  （JSON format）  Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{BODY} as array  Table 2‑16 Response parameter information( Column information)   |  |  | | --- | --- | | 列Key名 | Column name | | FILE\_NAME | Download file name | | FILE\_TYPE | File extension type  （EXCEL or SCSV） |   2) File information  （JSON format）  Is stored in Key {resultdata} -> Key{CONTENTS} -> Key{DOWNLOAD\_FILE}  **Table 2-17** **response parameter list**   |  |  | | --- | --- | | Column key name | Column name | | DOWNLOAD\_FILE | BASE64-encoded value of the downloaded file |   ・Response  The returned response is stored in JSON format   |  | | --- | | {  "status": "SUCCEED",  "resultdata": {  "CONTENTS": {  "BODY": {  "FILE\_NAME":Download file name  "FILE\_TYPE":"EXCEL" or "SCSV"  }  "DOWNLOAD\_FILE":BASE64-encoded value of the downloaded file  }  }  } |  UPLOAD\_SPREADSHEET(X-Command) Uploads the EXCEL file (or SCSV file) for registration and update of each menu via REST API.  Table 2‑18 HTTP header parameter list   |  |  | | --- | --- | | HTTP header | value | | X-Command | UPLOAD\_SPREADSHEET |  |  | | --- | | UPLOAD\_ SPREADSHEET　Json description example | | {  "FILE\_NAME":Upload file name  "UPLOAD\_FILE": BASE64-encoded value of the downloaded file  } |   ・Response  1) Record process results  （JSON format）  Is stored in Key{resultdata} -> Key{LIST} -> Key{NORMAL} -> Key{register、update、  delete、error} as array  Table 2‑19 Key Parameter list   |  |  |  | | --- | --- | --- | | **key** | **Value type** |  | | name | String | Process result type name | | ct | Number | Number of records (Per process result) |   2) Record process results  （JSON format）  The error information (relevant items and error details) is stored in Key {resultdata} -> Key{LIST} -> Key{ERROR\_DETAIL} as an array of lines,  similar to the file output when an error occurs after uploading an EXCEL file (or SCSV file) from the Web ( tableIUDByExcel\_exec\_YYYYMMDDhhmmss\_nnnnnnnnn.log) that is output when an error occurs after uploading an EXCEL file (or SCSV file) from the Web.  Table 2‑20 Response parameter list (Record information)   |  |  |  | | --- | --- | --- | | Column No. | Column data | Description | | 0 | Column 1 data |  | | 1 | Column 2 data |  | | … | … |  |   ・Response  The returned response is stored in JSON format   |  | | --- | | {  "status": "SUCCEED",  "resultdata": {  "LIST": {  "NORMAL": {  "register": {  "name": "Register",  "ct":  },  "update": {  "name": "Update",  "ct":  },  "delete": {  "name": "Abolish",  "ct":  },  "revive": {  "name": "Restore",  "ct":  },  "error": {  "name": "Error",  "ct":  }  },  "ERROR\_DETAIL": [  Line 10 MovementID1  Update Record was updated from a different session. Stopped the process. (Operation :Update)  ・・・（Abbr）・・・  ]  }  }  } |       Menu export / import |
| Import menu | 2100000212 |
| Excel bulk export | 2100000329 | 4. Excel bulk export/import |
| Excel bulk import | 2100000330 |
| 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 |
| Terraform | Execution | 2100080009 |
| Check operation status | 2100080010 |
| Management console | Version check | 2100000299 | 8 Check Version |
| Compare | Run Comparison | 2100190003 | 9 Comparison |

Table 1-2 Menus where RestAPI Cannot be used

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Menu create | Menu definition/creation | 2100160011 |
| Menu create | Menu create execution (Hidden by default) | 2100000205 |
| Basic console | ER diagram | 2100000326 |
| Terraform | Link terraform list | 2100080017 |

# Standard REST function

Users can use REST API REST API from external programs to operate the resources managed in ITA.  
The following shows the calling convention.

## Format of request

With ITA’s REST API, users will have to send HTTP requests to the different ITA Menu’s paths.

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 (:) |
| 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 Methods and Commands

As a general rule, we ask you to specify POST for the HTTP request method.

However, if the user is going to access a menu that has been set to require no authentication by ITA, GET can be specified as an exception.

Method

　　　　　├GET ‐　①

　　　　　｜

　　　　　└POST

(X-Command)｜

├【INFO】 ‐　②

　　　　　　　├【FILTER】 ‐　③

　　　　　　　├【FILTER\_DATAONLY】 ‐　④

　　　　　　　├【EDIT】 ‐　⑤

　　　　　　　├【LIST\_OPTIONS】 ‐　⑥

├【DOWNLOAD\_SPREADSHEET】‐　⑦

└【UPLOAD\_SPREADSHEET】 ‐　⑧

#### 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　   
 None

・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 2-2 Response parameter list (Column information)

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

1. Record information

(JSON format)(One array per Column (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 2-3 List of Response parameter (Record information)

|  |  |
| --- | --- |
| Column number | Column data |
| 0 | First data array |
| 1 | Second data array |
| … | … |

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

Table 2-4 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 2-5 HTTP Header parameter list

|  |  |
| --- | --- |
| HTTP header | Value |
| 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 2-6 HTTP header parameter list

|  |  |
| --- | --- |
| HTTP header | Value |
| 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.

If nothing is specified, the filter function will search for everything.

・NORMAL ‐ Normal LIKE search

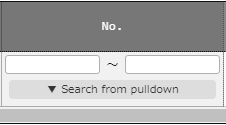
・RANGE ‐ Range search from level 1~5

LIST ‐ 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.

※Searching with Columns (e.g. item numbers) that specifies a number range in the GUI will produce the same search result, regardless if it is searched with “RANGE – START or with NORMAL.

Example）{"2":{"NORMAL":{"1"}}

{"2":{"RANGE":{"START":"1","END":""}}}



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 | | Column | 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. )　 FILTER parameter description (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 | … | | Column | 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. )　 FILTER parameter description(3) |
| 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"  }  }  } |

・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 2-7 Response parameter list (Column information)

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

1. Record information

(JSON format) (One array per Column (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 2-8 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.

　For details on how to search for Access permission roles, please refer to

“Exastro-ITA\_User\_instruction\_manual\_Role-based\_access\_control\_for\_data\_records".

#### EDIT(X-Command)

Register record or update/discard/restore existing record.

・HTTP Header

Table 2-9 HTTP header parameter list

|  |  |
| --- | --- |
| HTTP header | Value |
| 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.

When updating and you are abbreviating parameters, make sure to update without changing the value, even if the item you are changing is necessary. Make note that you cannot abbreviate unique items that identifies data (Such as ID and No.).

For column number 0, “execution process type”, please specify either “Register”, “Update”, “Discard”, “Update”.

|  |
| --- |
| **Example(1) 　Register** |
| Column 0 is “execution process type”, column 1 is “Discard” , column 2 is “ID” (primary key column)  ~ Abbr ~  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": "",  ・・・（Abbr）・・・  "10": "Remarks",  "11": "",  "12": "",  "13": ""  },  "1": {  "0": " Update ",  "1": "",  "2": "",  ・・・（Abbr）・・・  "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)  ~ Abbr ~  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)  ~ Abbr ~  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).

・Response

1. Execution result of each record

(JSON format)

Stored as array in key{resultdata} -> key{LIST} -> key{NORMAL} -> key{register、update、

delete、error}

Table 2-10 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 2-11 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・・・  ],  ・・・(Abbr)・・・  ]  }  }  } |
|
|

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. |

#### LIST\_OPTIONS(X-Command)

Acquires a list of values that can be selected from the pulldown selections from the different menus.

※The selection value of Access permission/Access permission roles cannot be acquired.

Table 2‑12 HTTP header parameter list

|  |  |
| --- | --- |
| HTTP header | value |
| X-Command | LIST\_OPTIONS |

・Response

1)Column information（number and name）

（JSON format）

Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{BODY} -> Key{0}As a key starting with the numeric value “0”

Table 2‑13 Response parameter list (Column information)

|  |  |
| --- | --- |
| Column number | Column name |
| 0 | Column 1 |
| 1 | Column 2 |
| … | … |

2)Record information

（JSON format）

Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{BODY} -> Key{1} as an array

Only data from pulldown items will be acquired. Other items will be blank.

Table 2‑14 Response parameter list (Record information)

|  |  |  |
| --- | --- | --- |
| Column number | Column data | Description |
| 0 | Column 1 data |  |
| 1 | Column 2 data |  |
| … | … |  |

・Response

The returned response is stored in JSON format.

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "CONTENTS": {  "BODY": {  0: {  0:"Execution process type"  1:・・・  2:・・・  },  1: {  0:{  0:"Register”  1:"Update”  2:"Abolish”  3:"Restore”  }  1:{Table if item is pulldown、Blank if otherwise}  ・・・  }  }  }  } |

DOWNLOAD\_SPREADSHEET(X-Command)

Download menus as excel files with the filter function active. If the menu exceeds the maximum excel line number, it will be downloaded as an SCSV file.   
The parameter specification is the same as FILTER

Table 2‑15 HTTP header parameter

|  |  |
| --- | --- |
| HTTP Header | Value |
| X-Command | DOWNLOAD\_SPREADSHEET |

・Response

1) Record information

（JSON format）

Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{BODY} as array

Table 2‑16 Response parameter information( Column information)

|  |  |
| --- | --- |
| 列Key名 | Column name |
| FILE\_NAME | Download file name |
| FILE\_TYPE | File extension type  （EXCEL or SCSV） |

2) File information

（JSON format）

Is stored in Key {resultdata} -> Key{CONTENTS} -> Key{DOWNLOAD\_FILE}

**Table 2-17** **response parameter list**

|  |  |
| --- | --- |
| Column key name | Column name |
| DOWNLOAD\_FILE | BASE64-encoded value of the downloaded file |

・Response

The returned response is stored in JSON format

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "CONTENTS": {  "BODY": {  "FILE\_NAME":Download file name  "FILE\_TYPE":"EXCEL" or "SCSV"  }  "DOWNLOAD\_FILE":BASE64-encoded value of the downloaded file  }  }  } |

#### UPLOAD\_SPREADSHEET(X-Command)

Uploads the EXCEL file (or SCSV file) for registration and update of each menu via REST API.

Table 2‑18 HTTP header parameter list

|  |  |
| --- | --- |
| HTTP header | value |
| X-Command | UPLOAD\_SPREADSHEET |

|  |
| --- |
| UPLOAD\_ SPREADSHEET　Json description example |
| {  "FILE\_NAME":Upload file name  "UPLOAD\_FILE": BASE64-encoded value of the downloaded file  } |

・Response

1) Record process results

（JSON format）

Is stored in Key{resultdata} -> Key{LIST} -> Key{NORMAL} -> Key{register、update、

delete、error} as array

Table 2‑19 Key Parameter list

|  |  |  |
| --- | --- | --- |
| **key** | **Value type** |  |
| name | String | Process result type name |
| ct | Number | Number of records (Per process result) |

2) Record process results

（JSON format）

The error information (relevant items and error details) is stored in Key {resultdata} -> Key{LIST} -> Key{ERROR\_DETAIL} as an array of lines,

similar to the file output when an error occurs after uploading an EXCEL file (or SCSV file) from the Web ( tableIUDByExcel\_exec\_YYYYMMDDhhmmss\_nnnnnnnnn.log) that is output when an error occurs after uploading an EXCEL file (or SCSV file) from the Web.

Table 2‑20 Response parameter list (Record information)

|  |  |  |
| --- | --- | --- |
| Column No. | Column data | Description |
| 0 | Column 1 data |  |
| 1 | Column 2 data |  |
| … | … |  |

・Response

The returned response is stored in JSON format

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "LIST": {  "NORMAL": {  "register": {  "name": "Register",  "ct":  },  "update": {  "name": "Update",  "ct":  },  "delete": {  "name": "Abolish",  "ct":  },  "revive": {  "name": "Restore",  "ct":  },  "error": {  "name": "Error",  "ct":  }  },  "ERROR\_DETAIL": [  Line 10 MovementID1  Update Record was updated from a different session. Stopped the process. (Operation :Update)  ・・・（Abbr）・・・  ]  }  }  } |

# 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 a HTTP request with the following information.

・Path

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

Please refer to “T**able 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 |
| 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"  },  {  ・・・(Abbr)・・・  }  ]  },  " Menu group ID ": {  ・・・(Abbr)・・・  }  }  }  } |

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 |
| Specified\_timestamp | Time specification Only input something when Mode no.2, Time specification is chosen.  The date format is as following:  YYYY-MM-DD H:i E.g) 2020-01-01 00:00 |

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

|  |
| --- |
| **Example) JSON description** |
| {  "2100000002": [  2100000202,  ・・・(Abbr)・・・  2100000222  ],  "2100000003": [  ・・・(Abbr)・・・  ],  “dp\_mode”:”1”,  “abolished\_type”:”1” “specified\_timestamp”:” 2020-01-01 00:00”  } |
|
|

・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 import menus with RestAPI.

The functions available are the 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 |
| 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 | Specifies the value of target file encoded in base64. |

|  |
| --- |
| 1）UPLOAD　Json description example |
| {  "zipfile":{  "name":"ita\_exportdata\_20191224092830.kym",  "base64":"・・・(Abbr)・・・"  }  } |

・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"  }  ・・・(Abbr)・・・  ],  " Menu group ID ": {  ・・・(Abbr)・・・  ],  "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 |
| Specified\_timestamp | The specified time stamp.  If the dp\_mode is set to 1, the value will be null. |
| 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

Start importing using the uploaded file.

Users can specify the target menu group, menu ID and 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": [  ・・・(Abbr)・・・  ],  "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 for 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 in bulks using Excel

## RestAPI for exporting in bulks using Excel

Users can export in bulks using Excel through RestAPI.  
The functions available are the same as the operation in the “Export menu” menu in the “Export/Import” menu group.

Table 4‑1 Menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu名 | MenuID |
| Export/Import | Excel bulk export | 2100000329 |

### Request format

Send HTTP request with the following information.

・Path

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

Please see “**Table 4-2　List of parameters that can be specified for X-Command”** for the MenuID

・HTTP header

Table 4‑2 HTTP Header parameter list

|  |  |
| --- | --- |
| HTTP header | Description |
| Content-Type | Specifies “application/json”。 |
| Authorization | When accessing a menu that needs ITA authentication, specify the “Login ID” and “Password”\* value. Make sure that they are connected using a half-width colon (:) and that they are encoded with base64. |
| X-Command | The following can be selected  EXECUTE  INFO |

Parameters that can be specified for X-Command

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

|  |  |  |  |
| --- | --- | --- | --- |
| X-Command | Description | Screen | MenuID |
| INFO | Obtains a list of menus that can be exported | Excel bulk export | 2100000329 |
| EXECUTE | Exports the specified MenuID's (pre-downloaded) zip file. | Excel bulk export | 2100000329 |

The following section explains the different X-command parameters.

### INFO

Outputs a list of menus that can be exported

・Parameter

　No parameters can be specified.

　・Response

　The returned response is stored with JSON format.

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "MENU\_LIST": {  "Menu groupID ": {  "menu\_group\_name": "Menu group name",  "menu": [  {  "menu\_id": "MenuID ",  "menu\_name": "Menuname"  },  {  ・・・（abbr）・・・  }  ]  },  "Menu groupID ": {  ・・・（abbr）・・・  }  }  }  } |

Table 4‑4　Response item list

|  |  |
| --- | --- |
| Item name | Remarks |
| Menu groupID | Constructs a Menu array with the Menu group ID as key. |
| menu\_group\_name | Menu group name |
| menu\_id | MenuID |
| menu\_name | Menu name |

### EXECUTE

Exports the specified Menu ID

・Parameter

　　Please specify the following as content in JSON format.

**Table 4‑5 Excel bulk export Parameter list**

|  |  |
| --- | --- |
| Parameter name | Setting value |
| Menu groupID | MenuID |
| abolished\_type | Abolished information  1:All records  2:Leave out abolished data  3:Only abolished data |

※The Menu groupID and MenuID are acquired from the INFO return value.

|  |
| --- |
| **Example）JSON description** |
| {  "2100000002": [  “2100000202”,  ・・・（Abbr）・・・  “2100000222”  ],  "2100000003": [  ・・・（Abbr）・・・  ],  “abolished\_type”: “3”  } |
|
|

・Response

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

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "TASK\_ID": "Task No ",  "RESULTCODE": "Result code",  "RESULTINFO": "Detailed information"  }  } |
|
|

Table 4‑6　Response item list

|  |  |
| --- | --- |
| Item name | Remarks |
| TASK\_ID | Task No  Users can check the run status by searching with the Task No in the "Excel bulk export/import list" menu. |
| RESULTCODE | Code that tells us the status of the command execution  000：Successfully executed  002：Could not execute |
| RESULTINFO | Detailed information |

## RestAPI for importing in bulks using Excel

Users can import in bulks using Excel through RestAPI.

The functions available are the same as the operation in “Import menu” menu in “Export/Import” menu group.

Table 4‑7 Menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | MenuID |
| Export/Import | Excel bulk export | 2100000330 |

### Request format

Send HTTP request with the following information.

・Path

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

Please see **“**エラー! 参照元が見つかりません。**” for the Menu ID.**

・HTTP header

**Table 4‑8　HTTP header Parameter list**

|  |  |
| --- | --- |
| HTTP header | Description |
| Content-Type | Specifies “application/json” |
| Authorization | When accessing a menu that needs ITA authentication, specify the “Login ID” and “Password”\* value. Make sure that they are connected using a half-width colon (:) and that they are encoded with base64. |
| X-Command | The following can be selected  UPLOAD  EXECUTE |

Parameters that can be specified for X-Command

Table 4‑9 List of parameters that can be specified for X-Command

|  |  |  |  |
| --- | --- | --- | --- |
| X-Command | Description | Screen | MenuID |
| UPLOAD | Uploads the exported zip file and outputs a list of menus the file can be imported to. | Excel bulk import | 2100000330 |
| EXECUTE | Imports the zip file to the selected Menu. | Excel bulk importト | 2100000330 |

The following section explains the different X-command parameters.

### UPLOAD

Upload the previously exported file.

The file is base64 encoded and transferred as a parameter.

・Parameter

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

**Table 4‑10　Excel bulk import UPLOAD Parameter list**

|  |  |
| --- | --- |
| Parameter name | Setting value |
| name | Target file name |
| base64 | Specifies the value of target file encoded in base64 |

|  |
| --- |
| 1）UPLOAD　Json |
| {  "zipfile":{  "name":"ITA\_FILES\_20191224092830.zip",  "base64":"・・・Abbr・・・"  }  } |

・Response

　　The returned response is stored in JSON format.

|  |
| --- |
| {  "status": ""SUCCEED",  "resultdata": {  "upload\_id": "Upload ID ",※  "data\_portability\_upload\_file\_name": "File name",  “abolished\_type”: “1”,  "IMPORT\_LIST": {  "Menu groupID ": {  "menu\_group\_name": "Menu group name",  "menu": [  {  "menu\_id": "MenuID ",  "menu\_name": "Menu name",  “file\_name”:”File name”  }  ・・・（Abbr）・・・  ],  "Menu groupID ": {  ・・・（Abbr）・・・  ],  "RESULTCODE": "Result code",  "RESULTINFO": "Detailed information"  }  },  "UNIMPORT\_LIST": {  "Menu groupID ": {  "menu\_group\_name": "Menu group name",  "menu": [  {  "menu\_id": "MenuID ",  "menu\_name": "Menu name",  “file\_name”:”File name”,  “error”:”Error contents”  }  ・・・（Abbr）・・・  ],  }  }  } |
|
|

※The upload ID is used when we are importing (EXECUTE).

Table 4‑11　Response items list

|  |  |
| --- | --- |
| Item name | remarks |
| upload\_id | Value assigned when the upload is successful.  Will be used later when running “EXECUTE”. |
| data\_portability\_upload\_file\_name | File name |
| abolished\_type | Abolished information  1: All records  2: Excluded abolished info.  3: Only abolished info. |
| IMPORT\_LIST | List of imported menus |
| UNIMPORT\_LIST | List of menus that could not be imported. |
| Menu group ID | Menu groupIDをKeyとしてMenuの配列を構成する |
| menu\_group\_name | Menu group name |
| menu\_id | Menu ID |
| menu\_name | Menu name |
| file\_name | File name |
| error | Error contents |
| RESULTCODE | コマンド実行の成否のコード  000：正常終了  002：実行不可 |
| RESULTINFO | Detailed information |

### EXECUTE

Use the uploaded file to start the import process.

Users can specify the target menu group and menu ID.

・Parameter

Specify the following to “content” in JSON format.。

**Table 4‑12** Excel bulk import **EXECUTE Parameter**

|  |  |  |
| --- | --- | --- |
| Parametername | Setting value | Remarks |
| Menu groupID | MenuID |  |
| upload\_id |  | This is the "UPLOAD" return value, but without the "A\_" prefix. |
| data\_portability\_upload\_file\_name | File name |  |

|  |
| --- |
| 1）EXECUTE　Json description |
| {  "2100070001": [  “2100070001”,  “2100070002”,  “2100070003”  ],  "2100020002": [  ・・・（Abbr）・・・  ],  "upload\_id": "A\_20191217090335772040239",※  "data\_portability\_upload\_file\_name": "ITA\_FILES\_20191213095733.zip"  } |

※Use the "upload\_id" value gathered from UPLOAD and put "A\_" in front of it.

・Response

　The return response is stored in JSON format. Please see the table below for information regarding the items.

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "TASK\_ID": "Excel bulk import execution task No",  "RESULTCODE": "Result code",  "RESULTINFO": "Detailed information"  }  } |
|
|

Table 4‑33　Response item list

|  |  |
| --- | --- |
| Item name | Remarks |
| TASK\_ID | Task No  Task Noで[Excel一括エクスポート・インポート管理]Menuを検索することにより、実行状況を確認でKeyます。 |
| RESULTCODE | コマンド実行の成否のコード  000：正常終了  002：実行不可 |
| RESULTINFO | Detailed information |

# Using Symphony

## RestAPI for registering operations for Symphony.

It is possible to use Symphony with RestAPI.

The functions available are the ones that corresponds to the Symphony Menu group’s “Symphony Class edit” menu.

Table 5-1 Menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Symphony | Symphony class edit | 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 4-3　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 |
| 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 | Obtains the Symphony class column information | Symphony class edit | 2100000306 |
| FILTER | Refers the records that matches the Symphony class parameters. | Symphony class edit | 2100000306 |
| EDIT | Registers the Symphony class. | Symphony class edit | 2100000306 |

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

### INFO

Obtains the Symphony class column information  
※For more details, please refer to Chapter 2, Standard REST Function – “INFO(X-Command)

### FILTER

Obtains the column information (column number and name) of the records that match the condition specified in the Parameter, as well as the number of Columns, the contents of the records, and the column information of all records with normal status(obsolete or active).  
※For more details, please refer to Chapter 2, Standard REST Function – “FILTER(X-Command)

### EDIT

Registers, Edits, abolishes or revives Symphony classes.

・HTTP header

Table 5‑4 HTTP header parameter list

|  |  |
| --- | --- |
| HTTP header | Value |
| Method | POST |
| X-Command | EDIT |

・Parameter

1. Specify format

For information regarding the parameter specification items for each execution type, refer to the following Parameter Specification Items.

When specifying "Update”,"Abolish", or "Revive" for Item number 7,

Please set the “Last modified date for update “obtained from the X-Command:FILTER.

This data prevents overtaking updates.  
The “Last modified date for update” starts with a “T”.

**Table 5‑5** **Symphony class parameter list**

|  |  |  |
| --- | --- | --- |
| **Item No.** | **Parameter name** | **Remarks** |
| 0 | Process type | Register, Edit, Abolish, Revive. |
| 2 | Symphony Class ID | Blank when registering |
| 3 | Symphony name |  |
| 4 | Description |  |
| 5 | Remarks |  |
| 7 | Last modified date for update | T\_XXXXXXXXXXXXXXXXXXXX |
| 9 | Movement details | Movement details  See the table below for more information (Table 4-6) |

**Table 5‑6** **Movement details list**

|  |  |  |
| --- | --- | --- |
| **Item no.** | **Parameter name** | **Remarks** |
| 0 | Orchestrator ID | Orchestrator ID  Please see the table below (4-8) |
| 1 | Movement ID | Movement ID  Please see the “Movement list” menu. |
| 2 | Temporary stop | OFF:Blank  ON:checkedValue |
| 3 | Description |  |
| 4 | Operation ID(Individually specified) | Operation ID  Please see the “Input operation” menu. |

**Table 5‑7** **Symphony Class parameter Movement details**

|  |
| --- |
| **Parameter specified items (Register/Update)** |
| "9": [  {  "0": "Orchestrator ID",  "1": "Movement ID",  "2": "Temporary stop(OFF:/ON:checkedValue)",  "3": "Description",  "4": "Operation(Individually specified)"  },  {  **///Add more if you want to run multiple Movements///**  }  ] |
|
|
|
|
|

Table 5‑8　Orchestrator ID Table

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

|  |
| --- |
| **Parameter specified items (Register/Update)** |
| {  "0": "Process type：<Register or Update>",  "2": "Symphony class ID ",  "3": "Symphony name",  "4": "Description",  "5": "Remarks",  "7": " Last modified date for update ",  "9": [  {  "0": "Orchestrator ID",  "1": "Movement ID",  "2": "Temporary stop (OFF:/ON:checkedValue)",  "3": "Description",  "4": "Operation (Individually specified)"  }  **///Add more if you want to run multiple Movements///**  ]  } |
|
|
|
|
|

|  |
| --- |
| **Parameter specified items (Abolish/Revive)** |
| {  "0": "Process type：<Abolish or Revive>",  "2": "Symphony class ID",  "7": " Last modified date for update "  } |
|
|
|

※Please set the “Last modified date for update “obtained from the X-Command: FILTER.

This data prevents overtaking updates.  
The “Last modified date for update” starts with a “T”.

|  |
| --- |
| **Example) JSON Description: When running multiple process types.** |
| [  {  "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": "Abolish",  "2": 2,  "7": "T\_20191224135437197447"  },  {  "0": "Revive",  "2": 4,  "7": "T\_20191224135449793941"  }  ] |
|
|

・Response

登録・更新のRAW出力

Table 4‑9 Keyパラメータ一覧

|  |  |  |
| --- | --- | --- |
| **key** | **値の型** |  |
| 0 | 文字列 | 結果コード(別Tableを参照) |
| 1 | 文字列 | 詳細コード(別Tableを参照) |
| 2 | 文字列 | SymphonyクラスID |
| 3 | 文字列 | エラーメッセージ |

各レコードの処理結果について、廃止・復活時のRAW出力は『標準REST機能の利用』-『EDIT(X-Command)』を参照。

## 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 execution | 2100000308 |
| Symphony 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 5-11　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 | Execution status codes  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 4-14　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

Users can use RestAPI to operate the Symphony function..

The functions available are same as the ones 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 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 |
| 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 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,  ・・・(Abbr) refer to the following ① for the items・・・  "FOCUS\_MOVEMENT": 1  },  "MOVEMENTS": [  {  "CLASS\_ITEM": {  "ORCHESTRATOR\_ID": "3",  ・・・(Abbr) refer to the following ② for the items・・・  "NEXT\_PENDING": "checkedValue"  },  "INS\_ITEM": {  "STATUS": "11",  ・・・(Abbr) 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 |
| 10 | Terraform |

Table 5-27 Mapping table of status ID during Movement instance execution

|  |  |
| --- | --- |
| ID | Status |
| 1 | Not executed |
| 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 |

## RestAPI for Symphony list

### Request format

Send HTTP request with the following information.

・Path

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

See “Table 5-28 List of parameters that can be specified for X-Command” for the different Menu IDs

・HTTP header

Table 5‑29 HTTP header parameter list

|  |  |
| --- | --- |
| HTTP header | Description |
| Content-Type | Specifies “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 | The following can be selected: INFO  FILTER  FILTER\_DATAONLY  DOWNLOAD |

Parameters that can be specified for X-Command

Table 5‑30　Parameters that can be specified for X-Command

|  |  |  |  |
| --- | --- | --- | --- |
| X-Command | Description | Target screen | Menu ID |
| INFO | Acquires Symphony class information | Symphony list | 2100000310 |
| FILTER  FILTER\_DATAONLY | References records that matches Symphony class parameters | Symphony list | 2100000310 |
| DOWNLOAD | Acquires result data and input data from the Symphony list. | Symphony list | 2100000310 |

The following explains the different X-command parameters.

### INFO

Acquires Symphony class column information.

※See ”Standard REST function”-“INFO(X-Command)” for more information

### FILTER・FILTER\_DATAONLY

Acquires either the column information of the records that matches the set conditions or column information, record contents and line numbers for all the records with normal status (Abolished or restoring).

If FILTER\_DATAONLY is specified, the column information will be shortened.

※See ”Standard REST function”-“INFO(X-Command)” for more information

### DOWNLOAD

Acquires a BASE64-encoded zip file containing a set of input data and a set of result data for records that match the conditions specified by the parameters.

**Table 5‑20** **Symphony class parameter list**

|  |  |
| --- | --- |
| **Item key name** | **Parameter name** |
| SYMPHONY\_INSTANCE\_NO | Comma-separated Synphony instance number desired record |

|  |
| --- |
| **Example）JSON description：Symphony Instance number 1 and 2** |
| {  "SYMPHONY\_INSTANCE\_NO": [  　 "1",  "2"  ]  } |
|
|
|
|
|

・Response

1) Record line number

（JSON format）

Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{RECORD\_LENGTH} as numeric value

2) Record information

（JSON format）（1 array per line1（Column key and Column data））

Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{BODY} -> Key{0} as an array whose keys are numbers starting from 0.

**Table 5‑21** **Response parameter list**

|  |  |
| --- | --- |
| Column key name | Column name |
| SYMPHONY\_INSTANCE\_NO | Symphony instance No |
| INPUT\_DATA | Input data file name |
| RESULT\_DATA | Result data file name |

3) File information

（JSON format）（1 array per line1（Column key and Column data）

Key{resultdata} -> Key{CONTENTS} -> Key{ DOWNLOAD\_FILE} -> Key{0} as an array whose keys are numbers starting from 0.

**Table 5‑21** **Response parameter list**

|  |  |
| --- | --- |
| Column Key name | Column name |
| Input data file name | BASE64-encoded value of the downloaded file |
| Result data file name | BASE64-encoded value of the downloaded file |

・Response

The returned response is stored in JSON format

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "CONTENTS": {  “RECORD\_LENGTH”:Acquired number  "BODY": {  "0":{  "SYMPHONY\_INSTANCE\_NO": Synphony Instance No  "INPUT\_DATA ":Input data file name  "RESULT\_DATA":Result data file name  },  ・・・Repeat for number of acquisitions・・・  }  "DOWNLOAD\_FILE:{  "0"{  "Input data file name": BASE64-encoded value of the downloaded file  "Result data file name": BASE64-encoded value of the downloaded file  }  ・・・Repeat for number of acquisitions・・・  }  }  }  } |

# Conductor

## RestAPI for registering Conductor operations

Users can use RestAPI to operate the Conductor function.

The functions available are the ones that are in the "Conductor class edit" menu in the "Conductor" menu group.

Table 6‑1 Target menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Conductor | Conductor class edit | 2100180003 |

### Request format.

Send an HTTP request with the information below

・Path

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

・HTTP header

Table 6‑2　HTTP header parameter list

|  |  |
| --- | --- |
| HTTP header | Description |
| Content-Type | Specifies “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 | The following can be selected  INFO  FILTER  FILTER\_DATAOLY  EDIT |

Parameters that can be specified for X-Command

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

|  |  |  |  |
| --- | --- | --- | --- |
| X-Command | Description | Screen | Menu ID |
| INFO | Collect only the conductor class column information. | Conductor class edit | 2100180003 |
| FILTER FILTER\_DATAONLY | BColumnses for records that matches the Conductor class parameters. | Conductor class edit | 2100180003 |
| EDIT | Registers Conductor class. | Conductor class edit | 2100180003 |

The following section explains the different X-Command parameters.

### INFO

Collects the Conductor class’ column information。

※See "Standard Rest function" - "Filter (X-Command)" for more information

|  |
| --- |
| **E.g.）INFO collecting** |
| {…  "INFO":{  "edittype": "Execution process type",  "conductor": {  "conductor\_name":"Conductor name",  "id":" Conductor class ID",  "note":"Description",  "LUT4U":"Last modified date for update",  "ACCESS\_AUTH":"Access permission/Access permission roles"  },  "config": {  "editorVersion":"Editor information",  "nodeNumber":" Nodenumber",  "terminalNumber":"Terminal number",  "edgeNumber":"Edge number"  },  "node": {  "h":"height",  "id":" Node class id",  "terminal":{  "case":"Case no",  "edge":" link Line",  "id":"Terminal class id",  "targetNode":" link Node"  "type":"type",  "condition":"Condition",  "x":"x-axis",  "y":"y-axis"  },  "type":"type",  "PATTERN\_ID":"Pattern id",  "ORCHESTRATOR\_ID":"Orchestrator id",  "Name":" Movement name",  "CALL\_CONDUCTOR\_ID":"Conductor class ID",  "CONDUCTOR\_NAME":"Conductor name",  "CALL\_SYMPHONY\_ID":"Symphony class ID",  "SYMPHONY\_NAME":"Symphony name",  "OPERATION\_NO\_IDBH:" Operation ID",  "SKIP\_FLAG":"Skip flag",  "OPERATION\_NAME":"Operation name",  "note":"Description",  "w":"Width",  "x":"X-axis",  "y":"Y-axis"  },  "line": {  "id":"Line id",  "type":"type",  "inNode":"Inward connection node",  "outTerminal":"Outward connection terminal "  "inTerminal":"Inward connection terminal ",  "outNode ":"Outward connection node "  }  }  } |
|
|

Please see table 6-4～6.9 for the contents of the different item keys.

**Table 6‑4** **Conductor class parameter list**

|  |  |  |
| --- | --- | --- |
| **Item key** | **Parameter name** | **Remarks** |
| edittype | Execution process type | Register/Update/Abolish/Restore |
| conductor | Conductor details | Conductor information  More information in Table5-5 |
| config | Config name | Config information  More information in Table5-6 |
| node | Node details | Node information  More information in Table5-7 |
| terminal | Terminal details | Terminal information  More information in Table5-8 |
| line | Line details | Line information  More information in Table5-9 |

**Table 6‑5** **Conductor details list**

|  |  |  |
| --- | --- | --- |
| **Item key** | **Parameter name** | **Remarks** |
| conductor\_name | Conductor name |  |
| id | Conductor class ID | When registering, run this blank |
| note | Description |  |
| LUT4U | Last modified date for update | T\_XXXXXXXXXXXXXXXXXXXX |
| ACCESS\_AUTH | Access permission/Permission access roles |  |

**Table 6‑6** **Config details list**

|  |  |  |
| --- | --- | --- |
| **Item key** | **Parameter name** | **Remarks** |
| editorVersion | Editor information |  |
| nodeNumber | Node number |  |
| terminalNumber | Terminal number |  |
| edgeNumber | Edge number |  |

**Table 6‑7** **Node details list**

|  |  |  |
| --- | --- | --- |
| **Item key** | **Parameter name** | **Remarks** |
| h | Height |  |
| id | Node class id |  |
| type | Type | start/end/movement/call/call\_s/pause/ conditional-branch/ parallel-branch/ parallel-merge |
| PATTERN\_ID | Pattern id |  |
| ORCHESTRATOR\_ID | Orchestrator id | Orchestrator ID  See **エラー! 参照元が見つかりません。** for the supported tables |
| Name | Movement name |  |
| CALL\_CONDUCTOR\_ID | Conductor class ID |  |
| CONDUCTOR\_NAME | Conductor name |  |
| CALL\_SYMPHONY\_ID | Symphony class ID |  |
| SYMPHONY\_NAME | Symphony name |  |
| OPERATION\_NO\_IDBH | Operation ID |  |
| SKIP\_FLAG | Skip flag |  |
| OPERATION\_NAME | Operation name |  |
| note | Description |  |
| w | Width |  |
| x | X-axis |  |
| y | Y-axis |  |

**Table 6‑8** **Terminal details list**

|  |  |  |
| --- | --- | --- |
| **Item key** | **Parameter name** | **Remarks** |
| case | Case no |  |
| edge | Node class id |  |
| id | Terminal class id |  |
| targetNode | Link node |  |
| type | Type | in/out |
| condition | Condition |  |
| x | X-axis |  |
| y | Y-axis |  |

**Table 6‑9** **Line details list**

|  |  |  |
| --- | --- | --- |
| **Item key** | **Parameter name** | **Remarks** |
| id | Line id |  |
| type | Type | egde |
| inNode | Inward connection node |  |
| outTerminal | Outward connection terminal |  |
| outTerminal | Inward connection terminal |  |
| outNode | Outward connection node |  |

Table 6‑4　Conductor execution status ID table

|  |  |
| --- | --- |
| ID | Status |
| 1 | Not executed |
| 2 | Not executed(reserved) |
| 3 | Executing |
| 4 | Executing(Extended) |
| 5 | Normal end |
| 6 | Emergency stop |
| 7 | Abnormal end |
| 8 | Unexpected error |
| 9 | Reservation deleted |
| 11 | Reported |

### FILTER・FILTER\_DATAONLY

Gathers only records and column information that fits the conditions set within the parameter as well as operations with normal status’s ("Abolished" or "Restoring") record lines, record contents and column information.  
If FILTER\_DATAONLY is specified, the column information will be abbreviated.

※See "Standard Rest function" - "Filter (X-Command)" for more information

See Table5-4～5.9 for information regarding the column information

Table 6‑11　Items that can be specified with Parameters

|  |  |
| --- | --- |
| Item key | Target item |
| conductor\_name | Conductor name |
| id | Conductor class ID |
| note | Description |
| ACCESS\_AUTH | Access permission/Access permission roles |

E.g.） If you want to collect a Record with Conductor class ID being "1":  
{"id":{"LIST":["1"]}}

If you want to collect a record where the Conductor class name contains the string  
  
"Conductor":  
{"conductor\_name":{"NORMAL":"Conductor"}}

If you want to collect records with Conductor class ID 1 to 3.

{"id":{"RANGE":{"START":"1","END":"3"}}}

|  |
| --- |
| **E.g.）FILTER collecting** |
| {…  {  "edittype": "Execution process type",  "disuse": "Abolish",  "conductor": {  "conductor\_name":"Conductor name",  "id":"Conductor class ID",  "note":"Description",  "LUT4U":" Last modified date for update",  "ACCESS\_AUTH":"Access permission/Access permission roles"  },  "config": {  "editorVersion":"Editor information",  "nodeNumber":" Node number",  "terminalNumber":"Terminal number",  "edgeNumber":"Edge number"  },  "node": {  "h":"Height",  "id":" Node class id",  "terminal":{  "case":"Case no",  "edge":" Link line",  "id":"Terminal class id",  "targetNode":"Link node"  "type":"Type",  "condition":"Condition",  "x":"X-axis",  "y":"Y-axis"  },  "type":"Type",  "PATTERN\_ID":"Pattern id",  "ORCHESTRATOR\_ID":"Orchestrator id",  "Name":" Movement name",  "CALL\_CONDUCTOR\_ID":"Conductor class ID",  "CONDUCTOR\_NAME":"Conductor name",  "CALL\_SYMPHONY\_ID":"Symphony class ID",  "SYMPHONY\_NAME":"Symphony name",  "OPERATION\_NO\_IDBH:" Operation ID",  "SKIP\_FLAG":"Skip flag",  "OPERATION\_NAME":"Operation name",  "note":"description",  "w":"Width",  "x":"X-axis",  "y":"Y-axis"  },  "line": {  "id":"Line id",  "type":"Type",  "inNode":" Inward connection node",  "outTerminal":"Outward connection terminal "  "inTerminal":"Inward connection terminal ",  "outNode ":"Outward connection node "  }  },  1:{  "edittype":null,  "disuse":"",  "conductor":{  conductor\_name:Conductor\_002,  id:7,  note:null,  LUT4U:T\_20210629121745463936,  ACCESS\_AUTH:""  },  "config:{  "editorVersion":"1.0.2",  nodeNumber:4,  terminalNumber:5,  edgeNumber:3  },  node-1:{  "h":"58",  "id":"node-1",  "terminal":{  "terminal-1":{  "case":"",  "edge":"line-1",  "id":"terminal-1",  "targetNode":"node-3",  "type":"out",  "condition":[],  "x":"7666",  "y":"8000"  }  }  "type":"start",  "PATTERN\_ID":"",  "ORCHESTRATOR\_ID":"",  "Name":"",  "CALL\_CONDUCTOR\_ID":"",  "CONDUCTOR\_NAME":"",  "CALL\_SYMPHONY\_ID":"",  "SYMPHONY\_NAME":"",  "OPERATION\_NO\_IDBH":"",  "SKIP\_FLAG":"",  "OPERATION\_NAME":"",  "note":null,  "w":"198",  "x":"7485",  "y":"7971"  },  node-2:{  "h":"58",  "id":"node-2",  "terminal":{  "terminal-2":{  "case":"",  "edge":"line-2",  "id":"terminal-2",  "targetNode":"node-3",  "type":"in",  "condition":[],  "x":"8334",  "y":"8000"  }  }  "type":"end",  "PATTERN\_ID":"",  "ORCHESTRATOR\_ID":"",  "Name":"",  "CALL\_CONDUCTOR\_ID":"",  "CONDUCTOR\_NAME":"",  "CALL\_SYMPHONY\_ID":"",  "SYMPHONY\_NAME":"",  "OPERATION\_NO\_IDBH":"",  "SKIP\_FLAG":"",  "OPERATION\_NAME":"",  "note":null,  "w":"198",  "x":"8317",  "y":"7971"  },  node-3:{  "h":"58",  "id":"node-3",  "terminal":{  "terminal-3":{  "case":"",  "edge":"line-1",  "id":"terminal-3",  "targetNode":"node-1",  "type":"in",  "condition":[],  "x":"7888",  "y":"8000"  },  "terminal-4":{  "case":"",  "edge":"line-2",  "id":"terminal-4",  "targetNode":"node-2",  "type":"out",  "condition":[],  "x":"8099",  "y":"8000"  }  }  "type":"movement",  "PATTERN\_ID":"1",  "ORCHESTRATOR\_ID":"3",  "Name":"move1",  "CALL\_CONDUCTOR\_ID":"",  "CONDUCTOR\_NAME":"",  "CALL\_SYMPHONY\_ID":"",  "SYMPHONY\_NAME":"",  "OPERATION\_NO\_IDBH":"",  "SKIP\_FLAG":"0",  "OPERATION\_NAME":"",  "note":null,  "w":"245",  "x":"7871",  "y":"7978"  },  line-1:{  "id":"line-1",  "type":"egde",  "inNode":"node-3",  "outTerminal":"terminal-1",  "inTerminal":"terminal-3",  "outNode":"node-1"  },  line-2:{  "id":"line-2",  "type":"egde",  "inTerminal":"terminal-2",  "outNode":"node-3",  "inNode":"node-2",  "outTerminal":"terminal-4"  }  }  } |
|
|

Please see the table below for the ORCHESTRATOR\_ID for the Node.

Table 6‑12　 ORCHESTRATOR\_ID(Orchestrator ID)Table

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

See the following table for the Node END\_TYPEs

Table 6‑5 END\_TYPE table

|  |  |
| --- | --- |
| ID | Status |
| 5 | Normal end |
| 7 | Abnormal end |
| 11 | Reported |

### EDIT

Registers, Edits, abolishes or revives Conductor classes.

・HTTP header

Table 6‑14 HTTP header list

|  |  |
| --- | --- |
| HTTP header | Value |
| Method | POST |
| X-Command | EDIT |

・Parameter

1. Specify format

For information regarding the parameter specification items for each of the execution types, refer to the following parameter specification items.

When you are specifying " Update", "Abolish", or "Restore" for the Item key LUT4U, make sure to set the "Last modified date for update" obtained from the X-Command: FILTER.

This data prevents overtaking updates.

The "Last modified date for update" starts with "T\_".

|  |
| --- |
| **Parameter specification items（Register/Update）** |
| {  "edittype": "Execution process type：<Register or Update>",  "conductor": {  "conductor\_name":"Conductor name",  "id":" Conductor class ID",  "note":"Description",  "LUT4U":"Last modified date for update",  "ACCESS\_AUTH":"Access permission/Access permission role"  },  "config": {  "editorVersion":"Editor information",  "nodeNumber":" Node number",  "terminalNumber":"Terminal number",  "edgeNumber":"Edge number"  },  "node-1": {  "h":"Height",  "id":" Node class id",  "terminal":{  "terminal-1": {  "case":"Case no",  "edge":" Link line",  "id":"Terminal class id",  "targetNode":"Link node"  "type":"type",  "condition":"状Condition",  "x":"X-axis",  "y":"Y-axis"  },  **・・・Make one for each terminal (terminal-1, terminal-2… terminal-n)・・・・**  "type":"Type",  "PATTERN\_ID":"Pattern id",  "ORCHESTRATOR\_ID":"Orchestrator id",  "Name":" Movement name",  "CALL\_CONDUCTOR\_ID":"Conductor class ID",  "CONDUCTOR\_NAME":"Conductor name",  "CALL\_SYMPHONY\_ID":"Symphony class ID",  "SYMPHONY\_NAME":"Symphony name",  "OPERATION\_NO\_IDBH:" Operation ID",  "SKIP\_FLAG":"Skip flag",  "OPERATION\_NAME":"Operation name",  "note":" description",  "w":"Width",  "x":"X-axis",  "y":"Y-axis"  },  **・・・Make one for each node (node-1,node-2…node-n)・・・・**  "line-1": {  "id":"Line id",  "type":"Type",  "inNode":"Inward connection node",  "outTerminal":"Outward connection terminal "  "inTerminal":"Inward connection terminal ",  "outNode ":"Outward connection node "  },  **・・・Make one for each line (line-1,line-2…line-n)・・・・**  } |
|
|
|
|
|

|  |
| --- |
| **Parameter specification item（Abolish/Restore）** |
| {  "edittype": "Execution process type：<Abolish or Restore>",  "conductor": {  "id":" Conductor class ID",  "LUT4U":"Last modified date for update"  },  } |
|
|
|

※When you are specifying " Update", "Abolish", or "Restore" for the Item key LUT4U, make sure to set the "Last modified date for update" obtained from the Method:GET, X-Command: FILTER.

This data prevents overtaking updates.

The "Last modified date for update" starts with "T\_".

|  |
| --- |
| **E.g.）JSON Description example：Multiple execution process types** |
| {"0":{  "edittype": "Register ",  "conductor": {  "conductor\_name":"Conductor\_001",  "note":null,  "ACCESS\_AUTH":null  },  "config": {  "editorVersion":"1.0.2",  "nodeNumber":4,  "terminalNumber":5,  "edgeNumber":3  },  "node-1": {  "h":58,  "id":" node-1",  "terminal":{  "terminal-1": {  "edge":" line-1",  "id":" terminal-1",  "targetNode":" node-3"  "type":"out",  "x":"7666",  "y":"8000"  },  "type":"start",  "note":null,  "w":198,  "x":7485,  "y":7971  },  "node-2": {  "h":58,  "id":" node-2",  "terminal":{  "terminal-2": {  "edge":" line-2",  "id":" terminal-2",  "targetNode":" node-3"  "type":"in",  "x":"8334",  "y":"8000"  },  "type":"end",  "note":null,  "w":198,  "x":8317,  "y":7971  },  "node-3": {  "h":58,  "id":" node-3",  "terminal":{  "terminal-3": {  "edge":" line-1",  "id":" terminal-3",  "targetNode":" node-1"  "type":"in",  "x":"7888",  "y":"8000"  },  "terminal-4": {  "edge":" line-2",  "id":" terminal-4",  "targetNode":" node-2"  "type":"out",  "x":"8099",  "y":"8000"  },  "type":"movement",  "PATTERN\_ID":"1",  "ORCHESTRATOR\_ID ":"3",  "Name ":"move1",  "OPERATION\_NO\_IDBH":null,  "SKIP\_FLAG ":"0",  "OPERATION\_NAME":"",  "note":null,  "w":198,  "x":8317,  "y":7971  },  "1":{  "edittype": "Abolish",  "conductor": {  "id":"2",  "LUT4U": "T\_20191224135449793941"  }  }  } |
|
|

Users can also save json files, edit, register and abolish them from the Conductor class edit screen with a web bColumnser.



・Response

Register/ Update RAW output

Table 6‑15 Key parameter list

|  |  |  |
| --- | --- | --- |
| **key** | **Value type** |  |
| 0 | String | Result code(See other table) |
| 1 | String | Details code(See other table) |
| 2 | String | Conductor class ID |
| 3 | String | Error message |

For information regarding the RAW output when abolishing/restoring records, see "Standard Rest function - INFO(X-command)".

## 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 6‑6 HTTP Header parameter list.

|  |  |
| --- | --- |
| HTTP header | Description |
| 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 6‑7　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 6‑8　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 6‑9　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**

|  |
| --- |
| 2）EXECUTE　Json Description example (Individually specifying Operation and Skips) |
| Conductor class ID is 1、Operation ID is 001、  The following example specifies movement (node-3)Operation ID 2001 and skips movement(node-4).  ▽Written in JSON Format  {  "CONDUCTOR\_CLASS\_NO": 1,  "OPERATION\_ID": 1001,  "OPTION":{  "node-3":{  "OPERATION\_ID":2001  },  "node-4":{  "SKIP\_FLAG":1  }}  } |

**Figure 6.1-2 EXECUTE　Json Description example (Individual specification)**

　・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 6‑10　 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 6‑11 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 6‑12 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 6‑13　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 6-14 List of parameters that can be specified for X-Command

・HTTP header

Table 6‑15 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 X-command.

Table 6‑16　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 6‑17　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 6‑18　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 6‑19　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 6‑21” |
| 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 6‑20　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 6‑23” |
| STATUS | Node status. More in the table below “Table 6‑24” |
| 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 6‑21　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 6‑22 Orchestra ID table

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

Table 6‑23 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 6‑24 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 |

## RestAPI for Conductor list

### Request format

Send HTTP request with the following information.

・Path

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

See “Table 6-25 List of parameters that can be specified for X-Command” for the different Menu IDs

・HTTP header

Table 6-33 HTTP header parameter list

|  |  |
| --- | --- |
| HTTP header | Description |
| Content-Type | Specifies “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 | The following can be selected: INFO  FILTER  FILTER\_DATAONLY  DOWNLOAD |

Parameters that can be specified for X-Command

Table 6-34　 Parameters that can be specified for X-Command

|  |  |  |  |
| --- | --- | --- | --- |
| X-Command | Description | Target screen | Menu ID |
| INFO | Acquires Conductor class information | Conductor list | 2100180006 |
| FILTER  FILTER\_DATAONLY | References records that matches Conductor class parameters | Conductor list | 2100180006 |
| DOWNLOAD | Acquires result data and input data from the Conductor list. | Conductor list | 2100180006 |

The following explains the different X-command parameters.

### INFO

Acquires Conductor class column information.

※See ”Standard REST function”-“INFO(X-Command)” for more information

### FILTER・FILTER\_DATAONLY

Acquires either the column information of the records that matches the set conditions or column information, record contents and line numbers for all the records with normal status (Abolished or restoring).

If FILTER\_DATAONLY is specified, the column information will be shortened.

※See ”Standard REST function”-“INFO(X-Command)” for more information

### DOWNLOAD

Acquires a BASE64-encoded zip file containing a set of input data and a set of result data for records that match the conditions specified by the parameters.

**Table 6-35** **Conductor Class parameter list**

|  |  |
| --- | --- |
| **Item key name** | **Parameter name** |
| CONDUCTOR\_INSTANCE\_NO | Comma-separated Conductor instance number of desired record |

|  |
| --- |
| **Example）JSON description：Conductor Instance number 1 and 2** |
| {  "CONDUCTOR\_INSTANCE\_NO": [  　 "1",  "2"  ]  } |
|
|
|
|
|

・Response

1) Record line number

（JSON format）

Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{RECORD\_LENGTH} as numeric value

2) Record information

（JSON format）（1 array per line1（Column key and Column data））

Is stored in Key{resultdata} -> Key{CONTENTS} -> Key{BODY} -> Key{0} as an array whose keys are numbers starting from 0.

**Table 6-36** **Response parameter list**

|  |  |
| --- | --- |
| Column key name | Column name |
| CONDUCTOR\_INSTANCE\_NO | Conductor instance No |
| INPUT\_DATA | Input data file name |
| RESULT\_DATA | Result data file name |

3) File information

（JSON format）（1 array per line1（Column key and Column data）

Key{resultdata} -> Key{CONTENTS} -> Key{ DOWNLOAD\_FILE} -> Key{0} as an array whose keys are numbers starting from 0.

**Table 6-37** **Response parameter list**

|  |  |
| --- | --- |
| 列Key名 | 列名 |
| Input data file name | BASE64-encoded value of the downloaded file |
| Result data file name | BASE64-encoded value of the downloaded file |

・Response

The returned response is stored in JSON format

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "CONTENTS": {  “RECORD\_LENGTH”:Acquired number  "BODY": {  "0":{  "CONDUCTOR\_INSTANCE\_NO": Conductor instance No  "INPUT\_DATA ":Input data file name  "RESULT\_DATA":Result data file name  },  ・・・Repeat for number of acquisitions・・・  }  "DOWNLOAD\_FILE:{  "0"{  "Input data file name": BASE64-encoded value of the downloaded file  "Result data file name": BASE64-encoded value of the downloaded file  }  ・・・Repeat for number of acquisitions・・・  }  }  }  } |

# 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"  }  } |
|
|

# Version confirmation

## RestAPI for checking version

It is possible to check the current running version of an ITA as well as what drivers it has through RestAPI.

The available function(s) are the same as the ones in the “Management console” -> “Version check” menu.

Table 8‑1 Menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Management console | Version check | 2100000299 |

### 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 6-3 List of parameters that can be specified for X-Command” for Menu ID.

・HTTP header

Table 8-2 HTTP header parameter list

|  |  |
| --- | --- |
| HTTP Header | Description |
| 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 | The following can be selected:  INFO |

### INFO

Acquires the version information from all the different ITA functions. Will only acquire information from installed functions.

Table 8‑3 HTTP header parameter list

|  |  |
| --- | --- |
| HTTP header | Value |
| X-Command | INFO |

・Response

1) Record information

（JSON format）

Current running ITA Version is stored in :Key{resultdata} -> Key{CONTENTS} -> Key{BODY} -> Key{version}。

Installed drivers are stored in Key{resultdata} -> Key{CONTENTS} -> Key{BODY} -> Key{installed\_driver} as an array.

・Response

The returned response is stored in JSON format

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "CONTENTS": {  "BODY": {  "version":Installed ITA Version  "installed\_driver":[  ・・・List of installed drivers・・・  ]  }  }  } |

# Compare function

## RestAPI for the Compare function.

It is possible to operate the Compare function through RestAPI.

The functions available are the ones that corresponds to the following menu group.

Table 9‑1 Compare function menu list

|  |  |  |
| --- | --- | --- |
| Menu group | Menu name | Menu ID |
| Compare | Compare execution | 2100190003 |

### 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 6-3 List of parameters that can be specified for X-Command” for Menu ID.

・HTTP Header

Table 9‑2 HTTP header parameter list

|  |  |
| --- | --- |
| HTTP header | Description |
| Content-Type | Specifies “application/json” |
| Authorization | When accessing a menu that needs ITA authentication, specify the value of “Login ID” and “Password”\* connected with half-width colon (:) then encoded with base64. |
| X-Command | COMPARE |

The following explains the different X-command parameters。

### COMPARE

Specify the Comparison definition and standard date to run the comparison.

Users can also specify if they want to filter the output content.(Output all data or data with differences対only, CSV format or Excel format)

・Parameter

　Specify the following as content in JSON format.

Table 9‑3 Compare function parameter list

|  |  |
| --- | --- |
| Parameter name | 設定値 |
| COMPARE\_ID | Comparison definition ID |
| BASE\_TIMESTAMP\_0 | Specifies Compare menu 1 standard date （YYYY/MM/DD ｔｔ:ｍｍ） |
| BASE\_TIMESTAMP\_1 | Specifies Compare menu 2 standard date （YYYY/MM/DD ｔｔ:ｍｍ） |
| HOST\_LIST | Device list ID divided by comma |
| FORMATTER\_ID | Compare result output format  1：CSV (default)  2：Excel |
| OUTPUT\_TYPE | Compare result output contents  1:Output all (Default)  2:Output differences only |

|  |
| --- |
| **Example）JSON Description example** |
| {  "COMPARE\_ID": "1",  "BASE\_TIMESTAMP\_0": "",  "BASE\_TIMESTAMP\_1": "",  "HOST\_LIST ": "1,2,3",  "FORMATTER\_ID": "1",  "OUTPUT\_TYPE": "1"  } |
|
|

・Response

1) Record information

（JSON format）

The following items are stored in Key{resultdata} -> Key{CONTENTS} -> as an array.

Table 9‑4　Response item list

|  |  |
| --- | --- |
| Item name | Remarks |
| RESULTCODE | Result code  000：Compared  001：No results |
| RESULTINFO | Detailed information |
| FILENAME | Comparison result output file name |
| FILE | BASE64-encoded value of the downloaded file |

　　The returned response is stored in JSON format

|  |
| --- |
| {  "status": "SUCCEED",  "resultdata": {  "RESULTCODE": "Result code",  "RESULTINFO": "Detailed information",  "FILENAME": "File name",  "FILE": "BASE64-encoded value of the Compare result output file"  }  } |

# 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. |