# ITA_User_Manual

Collect function

─Ver 1.6 ─

# Disclaimer



# Trademark


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

# Table of contents


Table of contents ..... 2
Introduction ..... 3
1 Collect function overview ..... 4
1.1 About the collect function ..... 4
1.1.1 Collect function overview diagram ..... 4
1.1.2 Collect function Data registration process overview diagram ..... 5
1.2 Parametersheets registration ..... 6
1.2.3 Collect function requirements ..... 6
2 Handling Directories, File structures and varibles in the Collect function. ..... 7
2.1 Collectable Directories and File structures. ..... 7
2.1.1 Collectable File formats ..... 7
2.1.2 Collectable Directory configuration ..... 7
2.2 Variable and variable types ..... 10
3 Collect function console menu ..... 11
3.1 Menu/Screen list ..... 11
4 Collect function user manual ..... 13
4.1 Work flow. ..... 13
4.1.1 Collect function work flow. ..... 13
5 Collect function operation explanation ..... 16
5.1 Ansible Common console ..... 16
5.1.1 Collection interface information ..... 16
5.1.2 Collection item value list ..... 18
5.2 Ansible-Legacy、Ansible-Pioneer、Ansible-LegacyRole Console ..... 21
5.2.1 Check Collection status ..... 21
5.3 BackYard contents ..... 24
5.3.1 Overview of the Parameter sheet registration process. ..... 24
6 Operation ..... 25
6.1 Maintenance ..... 25
6.2 Maintenance ..... 25
7 Appendix ..... 27
7.1 References ..... 27

## Introduction

This document explains the ITA Collect function and how to use it.

# 1 Collect function overview

This section explains the collect function.

## 1.1 About the collect function

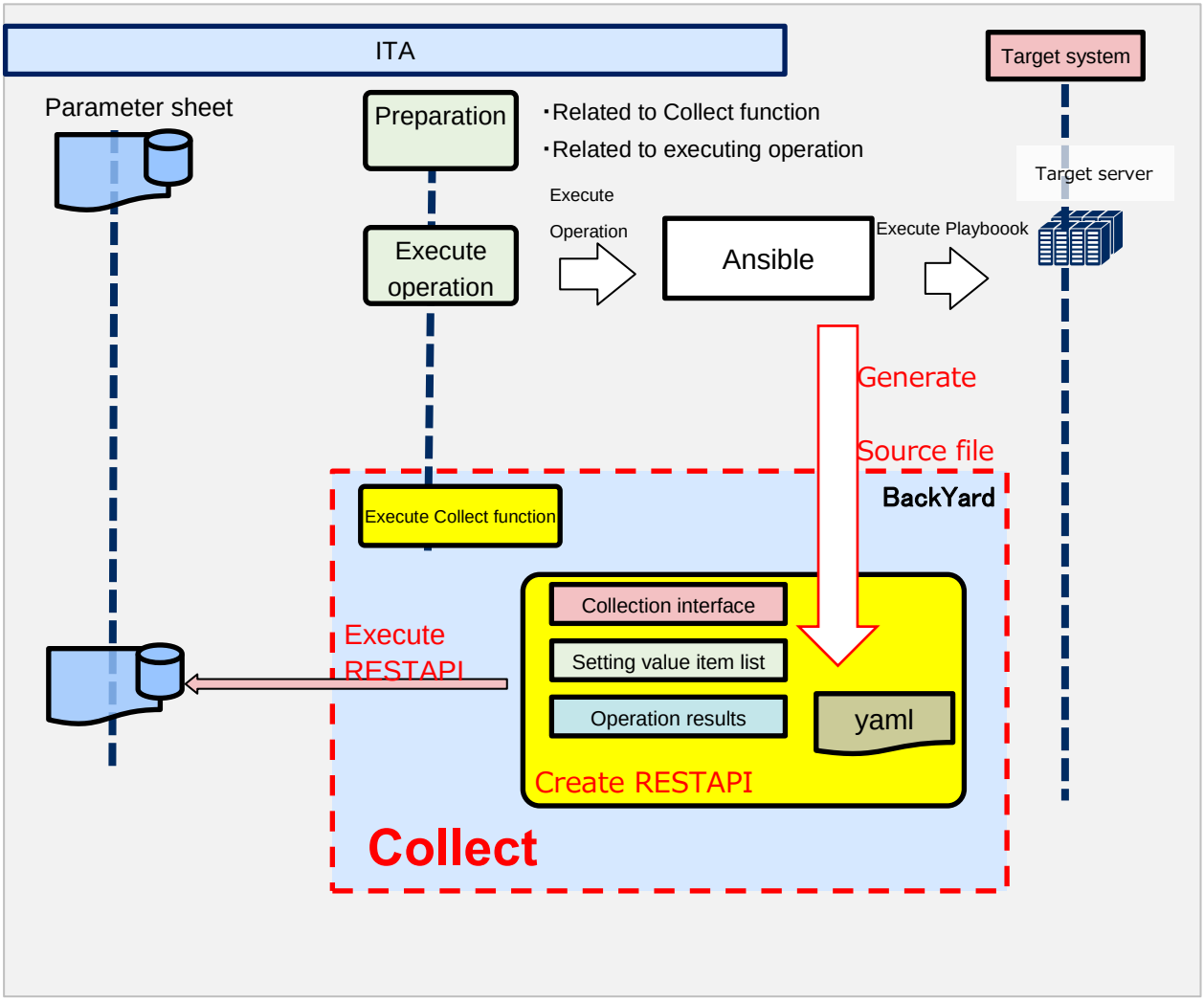
The collect function automatically registers values to parameter sheets. The values are based on the results of executed operations (source files output in a specified format) in ITA.

This function uses Ansible-Driver as target.

For more information about Ansible, please refer to the Ansible product manual
For more information about Ansible-Driver, please refer to “Exastro-ITA_User_Instruction_Manual_Ansible-Driver”
For more information about Parameter sheets, please refer to “Exastro-ITA_User_Instruction_Manual_Menu_creation_function”.

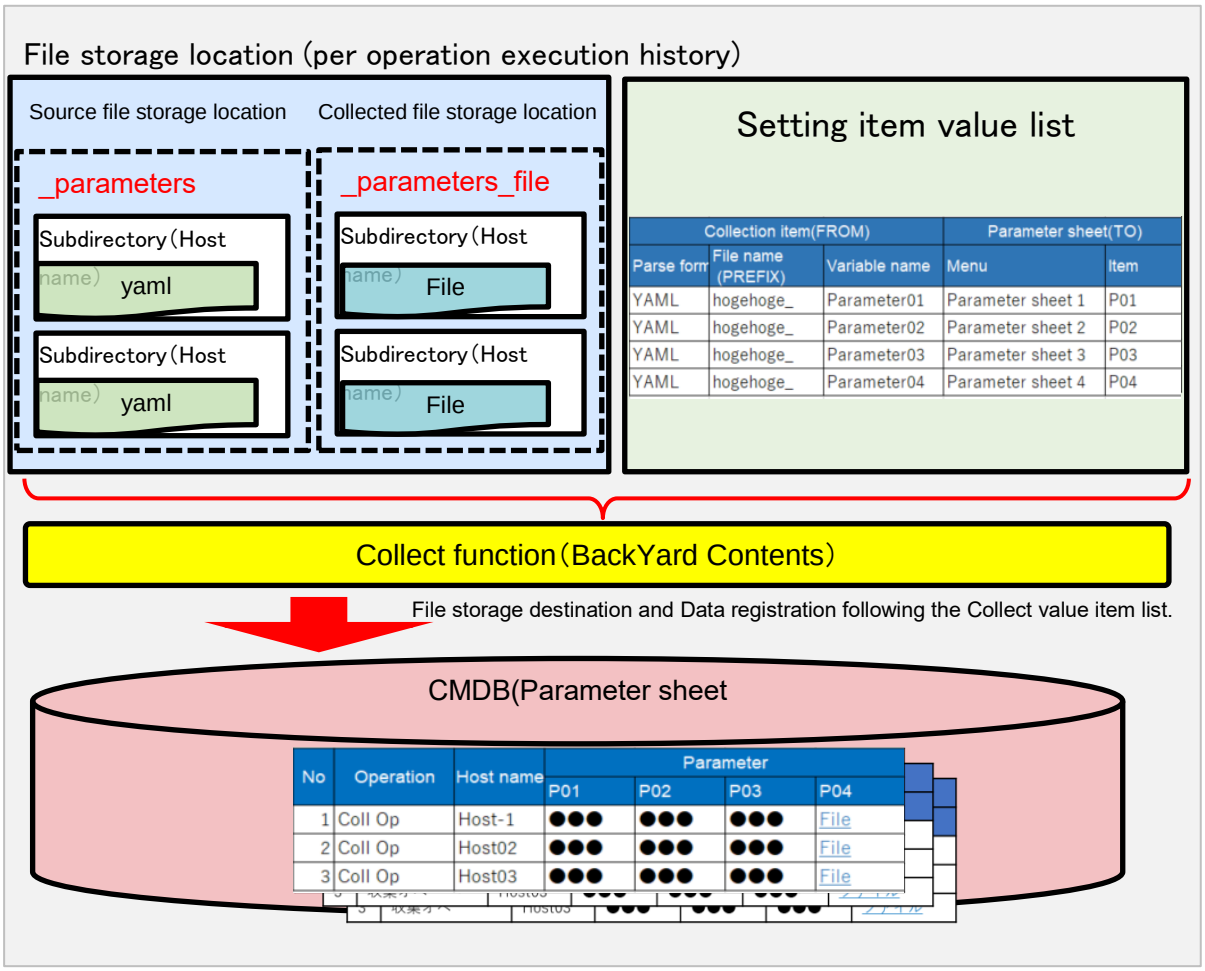
### 1.1.1 Collect function overview diagram

The following is a diagram that displays the entire process of using the Collect function.



### 1.1.2 **Collect function Data registration process overview diagram**

The following is a diagram of the Collect function Data registration process.



## 1.2 Parametersheets registration

The collect function is an option of ITA and uses ITA's standard REST API function for the Parameter sheet registration process

For more information about the REST API Function, please refer to "Exastro-ITA_User_Instruction_Manual_RESTAPI"

### 1.2.3 **Collect function requirements**

Make sure that the requirements below are met.

- ITA is installed with "Createparam" and "Ansible_driver" selected. (done in the installer)
- A parameter sheet (with Host/Operation) is created in the Menu definition/creation screen.
- The registration information (source file) is linked to the items in the Parameter sheet in the "Setting value item list"
- The Collection interfance information's REST access information is updated.
- The Collection target device (Host name) is already registered in the device list.

If the executed operations outputs any of the statuses below, it will be registered to the parameter sheet.

- Operation execution result, the operation has successfully ended.
- Directories and files are arranged in a specific structure as a result of the output of the operation execution.

※Each user must prepare the IaC(Plabook, Role) that generates source files going to be registered to the parameter sheets.

Reference：Ansible Playbook Collection（OS Setting collection）
https://github.com/exastro-suite/playbook-collection-docs/blob/master/README.ja.md

# 2 Handling Directories, File structures and varibles in the Collect function.

## 2.1 Collectable Directories and File structures.

### 2.1.1 Collectable File formats

(1) Files output in YAML format.

e.g.)
■File name：RH_snmp.yml
■File contents：

```
VAR_RH_sshd_config:
  - key: PermitRootLogin
    value: yes
  - key: PasswordAuthentication
    value: no
```

### 2.1.2 Collectable Directory configuration

The collectable directory path (output destination for the source file) can be handled as the following variable in IaC (Playbook, Role).

**Table 2-1 Collectable directory ITA Original variables**

| ITA original variable | Variable specified contents | Remarks |
|---|---|---|
| __parameter_dir__ | 「_parameters」 Operation result directory path | |
| __parameters_file_dir__ | 「_parameters_file」Operation result directory path | |
| __parameters_dir_for_epc__ | 「_parameters」Operation result directory path | |
| __parameters_file_dir_for_epc__ | 「_parameters_file」Operation result directory path | |

The upper directory of the collectable directories (parameters) depends on the "Data relay storage path (Ansible", Ansible driver execution mdoe and the No. of the operation.
(The "Data relay storage path (Ansible) can be found in Ansible Common -> Interface information in ITA.)

**Table 2-2 Collect function target Directory and file hierarchy**

| Hierarchy structure | Remarks |
|---|---|
| 【Upper directory】<br>\|- _parameters ※1<br>\| \|- localhost ※2<br>\| \|- SAMPLE.yml ※3<br>\|- _parameters_file ※4<br>\| \|- localhost ※2<br>\|- test.txt ※5 | ※1 Collectable directory（Fixed name）<br>※2Host name<br>（Items registered in the device list are collectable）<br>※3 Collectable file<br>※4Collectable directory for file uploads（Fixed name）<br>※5 Uploadable file |

※Hierarchical structure after data relay storage path (Ansible)

When creating a playbook that generate source files, not using the “Table 2-1 Collectable directory ITA Original variables”” for the output destination will require the user to write the Playbook with the following structure in mind.

**Table 2-3　Upper directory paths for the different Ansible-Driver modes**

| Mode | Mode identifier | Hierarchy structure | Remarks |
|---|---|---|---|
| Ansible-Legacy | legacy/ns/ | /DataRelayStoragePath(Ansible）/legacy/ns/ | |
| Ansible-Pioneer | pioneer/ns/ | / DataRelayStoragePath(Ansible）/pioneer /ns/ | |
| Ansible-LegacyRole | legacy/rl/ | / DataRelayStoragePath(Ansible）/legacy/rl/ | |

e.g.)　Collectable file paths and directory structures
Execution mode：Ansible-Legacy
Operation No　：1
Target host：localhost
Operation execution directory；/DataRelayStoragePath (Ansible）/legacy/ns/0000000001/in/
Operation results directory；/DataRelayStoragePath (Ansible）/legacy/ns/0000000001/out/

Collectable file path and directory structures：
/ DataRelayStoragePath (Ansible）/legacy/ns/0000000001/in/_parameters/localhost/SAMPLE.yml
/ DataRelayStoragePath (Ansible）/legacy/ns/0000000001/in/_parameters/localhost/OS/RH_snmpd.yml
/ DataRelayStoragePath (Ansible）/legacy/ns/0000000001/in/_parameters_file/localhost/TEST.txt

Or,

/ DataRelayStoragePath (Ansible）/legacy/ns/0000000001/out/_parameters/localhost/SAMPLE.yml
/ DataRelayStoragePath (Ansible）/legacy/ns/0000000001/out/_parameters/localhost/OS/RH_snmpd.yml
/ DataRelayStoragePath (Ansible）/legacy/ns/0000000001/out/_parameters_file/localhost/TEST.txt

If the user wants the file upload menu to be collectable, a file with the same name as the value of the source file variable (file name) must be placed under _parameters_.

For more information about Collection item value list settings, please refer to “5.1.2 Collection item value list”

As the maximum file size for uploads depends on the server specifications, please refer to “Exastro-ITA_User_Instruction_Manual_RESTAPI” for more details.

e.g.) Directory structure and source file contents when using variables of Normal variable structure.

■Structure

【Upper directory】

```
        |-  _parameters
        |    |-  localhost
        |         |-  SAMPLE.yml        ※Source file
        |-  _parameters_file
        |    |-  localhost
                  |-  test.txt                  ※Uploadable file
```

■Collectable file name：SAMPLE.yml

■File contents

```
VAR_upload_file: test.txt
```

## 2.2 Variable and variable types

The following 3 types of variables can be handled in the Collect function source file.

**Table 2.1 Variables and types**

| Type | Contents | Remarks |
|---|---|---|
| Normal variable | Can have one specific value defined per each variable name.<br>e.g.)<br>VAR_users: root | |
| Multiple specific value variable | Can have multiple specific values defined per each variable name<br>e.g.)<br>VAR_users:<br>- root<br>- mysql | |
| Multistage variable | Hierarchical variable.<br>e.g.)<br>VAR_users:<br>- user-name: alice<br>authorized: password } Member<br><br>Member variable names can contain any ascii character excluding the seven characters below. ( ‘0x20~0x7e can be used)<br>" . [ ] ' ¥ :<br>Keep in mind that there are a few characters that can’t be used at the beginning of a variable name unless they are enclosed in quotation marks.For more information, please refer to. | |

# 3 Collect function console menu

This section explains the ITA Console menu structure
For more information on how to log in to the web console and the basic operations/components of the menu screen, please refer to “Exastro-ITA_First_Step_Guide”

## 3.1 Menu/Screen list

### ① Ansible common console menu

The Ansible common console menu list is as following.

**Table 3-1 Common console Menu/screen list**

| No | Menu group | Menu/Screen | Description |
|---|---|---|---|
| 1 | Ansible common console | Collection interface information | Manage the connection interface information to the server that accesses the ITA standard REST Function.<br>The REST function is used when registering data to parameter sheets. |
| 2 | | Collected item value list | Set up the connection between the executed operation output results (Source file) and the parameter sheet items and manages the Collection function parameter sheets. |

### ② Ansible console menu

The list of menus corresponding to the Ansible consoles are as written below.

**Table 3-2 Ansible driver console Menu/Screen list**

| No | Menu group | | | Menu/Screen | Description |
|---|---|---|---|---|---|
| | Ansible Console | | | | |
| | Legacy | Legacy Role | Pioneer | | |
| 14 | ○ | ○ | ○ | Execution list | Manages operation execution history.<br>Refers to the registration status of the parameter sheet and execution log by the Collect function. |

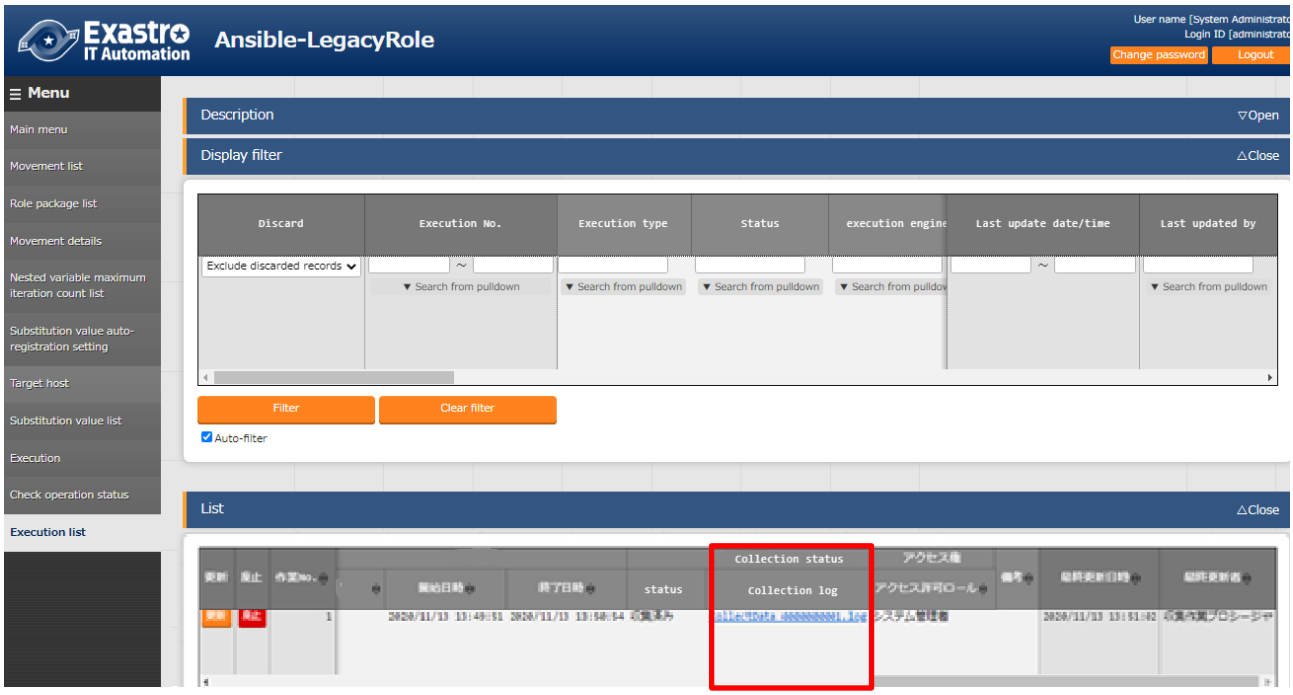



**Figure 3.1-1 Execution list screen**

# 4 Collect function user manual

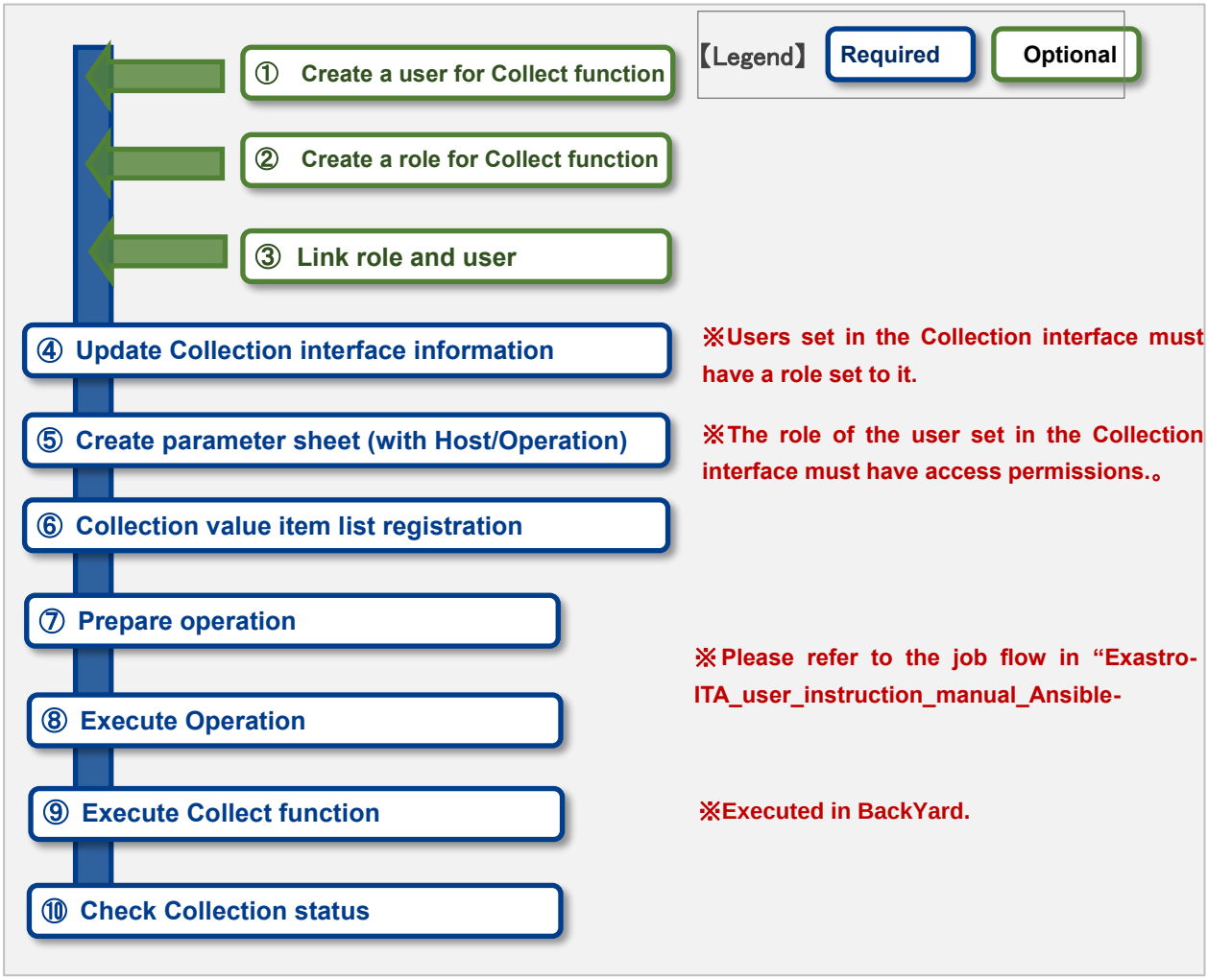
This section describes the how to use the Collect function.

## 4.1 Work flow.

The standard workflow for implementing the Colelct function is as following
For details on how to use ITA Ansible-Driver, please refer to Exastro-ITA_User_Instruction_Manual_Ansible-driver”
For details on how to use ITA Basic console, please refer to Exastro-ITA_User_Instruction_Manual_Basic_console”

### 4.1.1 Collect function work flow.

The following is the process before using Ansible-Legacy



- **Workflow and references.**

① **Create a user for the Collect function.**
Register a user for the Collect function in the ITA Management Console - Device list screen.
For details on how to register, please refer to “Exastro-ITA_User_Instruction_Manual_Management_console.”

② **Create a role for the Collect function**
Register a role for the Collect function in the ITA Management Console – Role list screen
For details on how to register, please refer to “Exastro-ITA_User_Instruction_Manual_Management_console.”

③ **Link role and user**
Link the role and user in the ITA Management console – Role/User link screen
For details, please refer to “Exastro-ITA_User_Instruction_Manual_Management_console.”

④ **Register Collection interface information**
Register the connection information in the Ansible Common console – Collection interface information screen
For details, please refer to “5.1.1 Collection interface information”

⑤ **Create Parameter sheet (with host/operation)**
Create a parameter sheet in the Menu creation console – Menu definition/creation screen
For details, pelase refer to “Exastro-ITA_User_Instruction_Manual_Menu_creation_function”

⑥ **Register Collection item value list.**
Register the information that links the source files to the items in the parameter sheet.
(Ansible common console – Collection item value list screen)
Fore details, please refer to “5.1.2 Collection item value list”.

⑦ **Prepare Operation**
Prepare the Operation to be executed.
For details, please refer to “Exastro-ITA_User_Instruction_Manual_Ansible-Driver”, “Exastro-ITA_User_Instruction_Manual_Symphony” and “Exastro-ITA_User_Instruction_Manual_Conductor”.

⑧ **Execute Operation**
Select the execution date/time, input operation, movement and workflow, and start the execution process.
For details regarding execution, please refer to “Exastro-ITA_User_Instruction_Manual_Ansible-Driver”, “Exastro-ITA_User_Instruction_Manual_Symphony” and “Exastro-ITA_User_Instruction_Manual_Conductor”.

⑨ **Execute Collect function**
Initiate the Parameter sheet registration process with the executed operation’s operation No. as target for the Collect function.
For details, please refer to “5.3 BackYard contents”.

⑩ **Check Collection status**

Ain the operation list screen, (Ansible-Legacy/ Ansible-Pioneer/Ansible-LegacyRole), users can check the Collection status of completed operations and download the log file(s).
For details, please refer to “5.2.1 Check Collection status”

# 5 Collect function operation explanation

This section explains how to operate the Collect function.

For details on how to register, please refer to “Exastro-ITA_User_Instruction_Manual_Basic_console”

## 5.1 Ansible Common console

This section explains how to operate the Ansible Common console.

### 5.1.1 Collection interface information

(1) Since the ITA’s standard REST API is used in this menu, it is required to update the Connection interface information for RESTAPI.

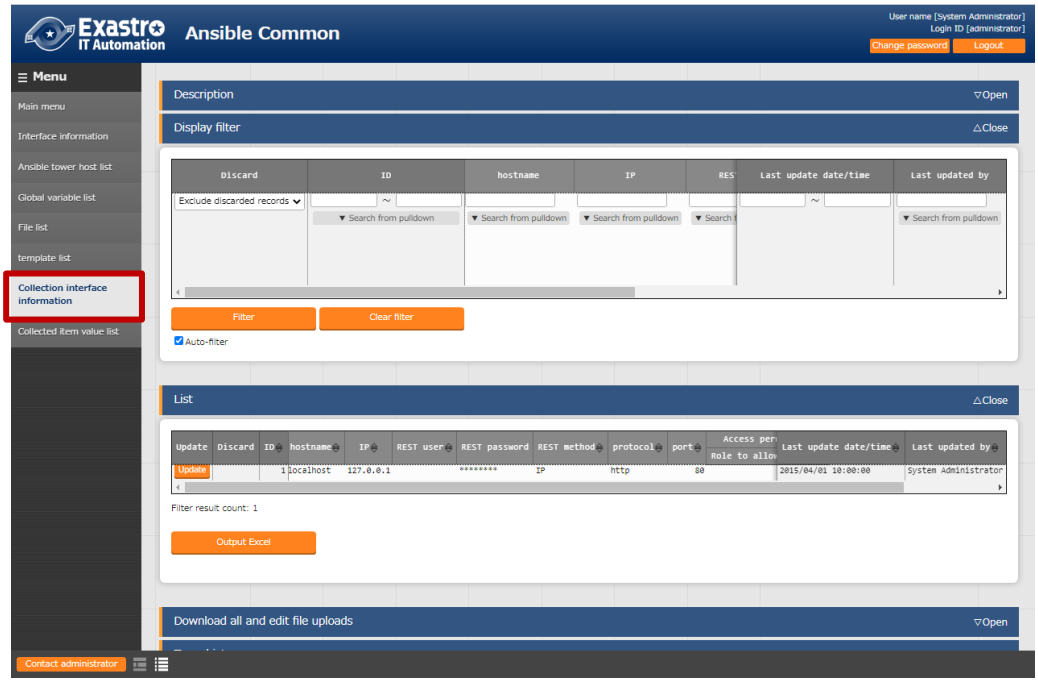



**Figure 5.1-1 Submenu screen（Collection interface information）**

(2) Register Collection interface information with the “List”-“Update” button.


**Figure 5.1-2 Update screen（Collection interface information）**

(3) The item list for the Collection interface information is shown below.
If the operation was executed with no Collection interface information registered or with multiple records registered, the Collect function will not register any information to the

Parameter sheet.

**Table 5.1-1 Registration screen, Item list（Interface information）**

| Item | Description | Input required | Input method | Constraints |
|---|---|---|---|---|
| | | | | |
| Host name | Input host name<br>Initial value：localhost | ○ | Manual input | |
| IP | Input IP Address<br>Initial value：127.0.0.1 | ○ | Manual input | |
| REST user | Input ITA user login ID | | Manual input | ※1 |
| REST password | Input ITA user login password | | Manual input | |
| RESTmethod | Choose IP or Host name<br>● IP<br>● Host name | ○ | Choose from list | |
| Protocol | Input protocol<br>Initial value：http | ○ | Manual input | |
| Port | Input port<br>Initial value：80 | ○ | Manual input | |
| Remarks | Free description field | - | Manual input | |

※1 Users entered in the “REST user” field will have the following required.

- The role that the user belongs to has to have permission to access the menu items in the created parameter sheet.
- The role linked to the user (in the Menu’s role information) has to be “Can Maintain” set to it.

For more information regarding Users, creating Roles and linking them, please refer to “Exastro-ITA_User_Instruction_Manual_Management_console.”

### 5.1.2 **Collection item value list**

(1) In the “Collection item value list”, set the link between the Colelction items and the items in the parameter sheet.

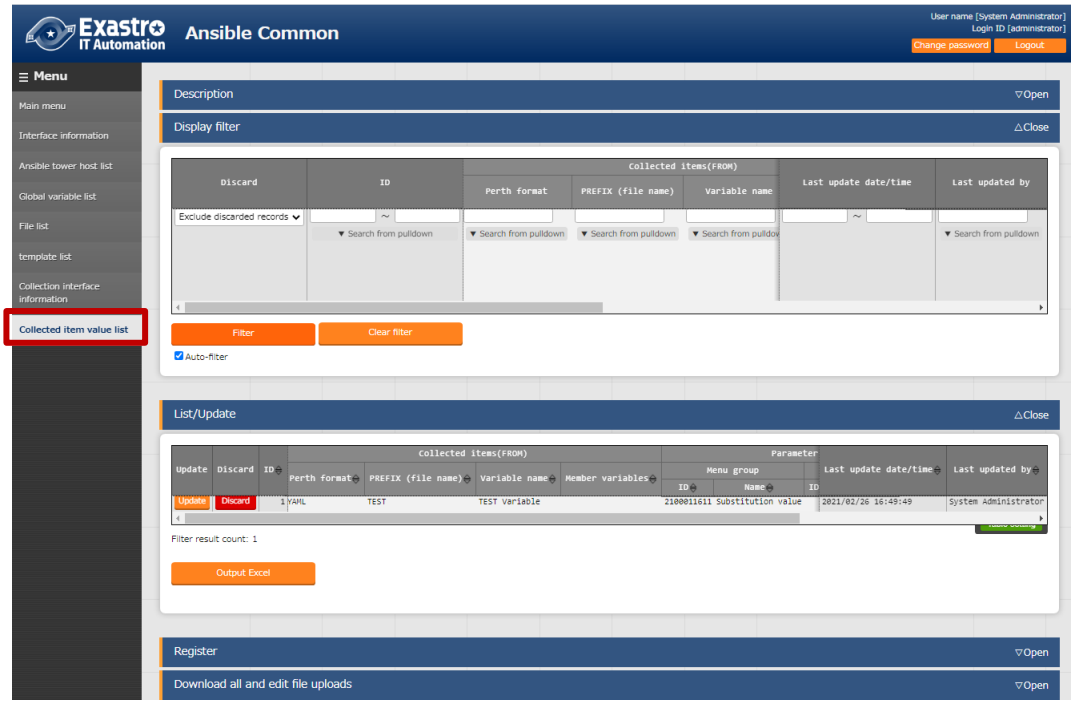



**Figure 5.1-3 Submenu screen（Collection item value list）**

(2) Register Collection item(s) with the “List“-“Start Registration” button.

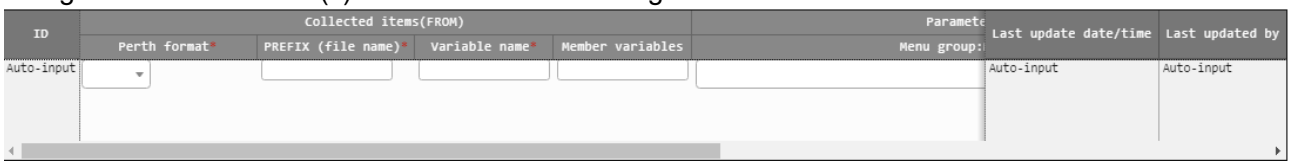



**Figure 5.1-4 Registration screen（Collection item value list）**

(3) The Collection item value list screen’s item list is as follows.

**Table 5.1-1 Registration screen Item list（Collection item value list）**

| Item | | Description | Input required | Input method | Constraints |
|---|---|---|---|---|---|
| | | | | | |
| Collected items (FROM) | Parse format | Select source file format. | ○ | Select from list | |
| | PREFIX(File name) | Enter the file name of the source file (Exclude the file extension). | ○ | Manual input | ※1 |
| | Variable name | Input variable name | ○ | Manual input | ※1 |
| | Member variables | Input if the variable is a multilevel | | Manual | ※1 |

| Item | | Description | Input required | Input method | Constraints |
|---|---|---|---|---|---|
| | | | | | |
| | | variable or if it has multiple concrete valies. | | input | |
| Parameter sheet(TO) | Menu group<br>Menu | Select from a list of menus created by the Menu creation function<br>Group name：Menu name | ○ | Select from list | |
| | Item | Select item. | ○ | Select from list | |

※1 Example of file name, variable and member value input value

e.g.) If the variable has a normal variable structure.
■File name：SAMPLE.yml
■File contents

VAR_sample_config_1: yes
VAR_sample_config_2: test_parameter

■Values that can be input in the Collected item (from) in the Collected value item list.
PREFIX(File name)： SAMPLE
Variable name： VAR_sample_config_1
VAR_sample_config_2

e.g.) If the variable has a multiple variable structure.
■File name：SAMPLE_2.yml
■File contents

VAR_sample2_conf:
  SAMPLE1
  SAMPLE2
  SAMPLE3
■Values that can be input in the Collected item(from) in the Collected value item list.
PREFIX(File name)： SAMPLE_2
Variable name： VAR_sample2_conf
Member variables： [0]
[1]
[2]

e.g.) If the variables has Multiple specific value structure.
■File name：RH_sshd.yml
■File contents

```
VAR_RH_sshd_config:
  - key: PermitRootLogin
    value: yes
  - key: PasswordAuthentication
    value: no
```

■Values that can be input in the Collected item(from) in the Collected value item list.

| | |
|---|---|
| PREFIX(File name)： | RH_sshd |
| Variable name： | VAR_RH_sshd_config: |
| Member variables： | [0].key |
| | [0].value |
| | [1].key |
| | [1].value |

e.g.) If the variable has Multiple specific value structure 2
■File name：RH_snmp.yml
■File contents

```
VAR_RH_snmpd_info:
  com2sec:
    - sec_name: "testsec"
      source: "192.168.1.0/24"
      community: "public"
    - sec_name: "local"
      source: "localhost"
      community: "private"
```

■Values that can be input in the Collected item(from) in the Collected value item list.

| | |
|---|---|
| PREFIX(File name)： | RH_snmp |
| Variable name： | VAR_RH_snmp_config: |
| Member variables： | com2sec[0].sec_name |
| | com2sec[0].source |
| | com2sec[0].community |
| | com2sec[1].sec_name] |
| | com2sec[1].source |
| | com2sec[1].community |

## 5.2 Ansible-Legacy、Ansible-Pioneer、Ansible-LegacyRole Console

### 5.2.1 Check Collection status

It is possible to check the status of completed operations and download the log files in each console's (Ansible-Legacy/ Ansible-Pioneer/Ansible-Legacy role) Execution list screen.

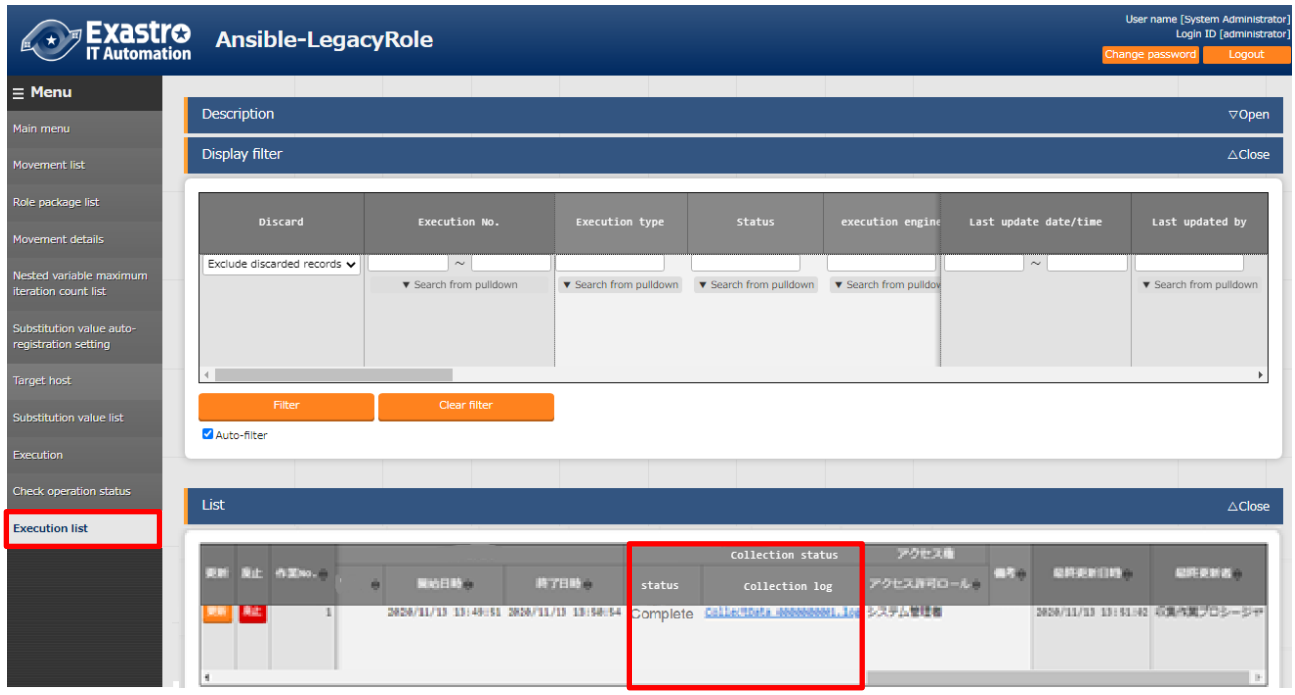



**Figure 5.2-1 Execution list screen**

**Table 5.2-1 Execution list Collection status details**

| Item | Description | Remarks |
|---|---|---|
| Status | Collection function execution status<br>Not target; Not a Collect function target (No target file)<br>Collected: Collect function executed.<br>Collected (with notification): If any errors occurred during registration/update. | ※ |
| Collection log | Download the collect function execution log. | |

**Table 5.2-2 Collection status details**

| Operation status | | Collect function target | Collection status | | |
|---|---|---|---|---|---|
| Status | Target file | | Status | Collection log | Remarks |
| Other than Complete | No | Not target | Blank | Blank | |
| Other than | Yes | Not target | Blank | Blank | |

| Operation status | | Collect function target | Collection status | | |
|---|---|---|---|---|---|
| Status | Target file | | Status | Collection log | Remarks |
| Complete | | | | | |
| Complete | No | Target | Not target | Blank | |
| Complete | Yes | Target | Collected | With log file | |
| Complete | No | Target | Collected (with notification) | With log file | |

※Regarding Status notations

- If the Operation status shows “Not complete”, the collection status will not be updated because it is not subject to the Collect function. Therefore, it will remain as “Blank”
- If the operation status shows as “Complete” and there are no files to collect, the status will show as “Collected” and the collection log will be blank.
- Even if the RESTAPI registration process fails during collecting the “ Collection interface information”, Settings item value list” or “ Menu access rights/permission roles”, the collection will show as “ Complete (with notification)”.

Example of Log file output contents.

e.g.)Example of Log file output contents (Registration process succeeded)

```
2020-11-13 13:51:02 Collect START ( Host name:ita-sample File name:RH_snmpd )
2020-11-13 13:51:02 REST DATA ( Host name: ita-sample Menu ID: 0000000004 Operation NO: 1 )
Array
(
    [0] => http://127.0.0.1:80/default/menu/07_rest_api_ver1.php?no=0000000004
    [1] => [["更新","","3","ita-sample","","","","","","2023¥/10¥/26 16:35_1:OP001","Root <root@localhost>
(configure ¥/etc¥/snmp¥/snmp.local.conf)","Unknown (edit
¥/etc¥/snmp¥/snmpd.conf)","public","notConfigUser","","","T_20201111115557819037",""]]
    [2] =>
{"status":"SUCCEED","resultdata":{"LIST":{"NORMAL":{"register":{"name":"¥u767b¥u9332","ct":0},"update
":{"name":"¥u66f4¥u65b0","ct":1},"delete":{"name":"¥u5ec3¥u6b62","ct":0},"revive":{"name":"¥u5fa9¥u6d3
b","ct":0},"error":{"name":"¥u30a8¥u30e9¥u30fc","ct":0}},"RAW":[["000","200",""]]}}}
)
2020-11-13 13:51:02 Collect END ( Host name:ita-sample File name:RH_snmpd )
```

e.g.) Example of Log file output contents (Registration process failed)

```
2020-11-06 13:32:52 Collect START ( Host name:ita-sample File name:RH_snmpd )
2020-11-06 13:32:52 [処理]REST Access failed
Array
(
    [0] => http://127.0.0.1:80/default/menu/07_rest_api_ver1.php?no=0000000005
    [1] => [["Register","","","ita-sample","","","","","","2023¥/10¥/26 16:35_1:OP001","Root
<root@localhost>(configure ¥/etc¥/snmp¥/snmp.local.conf)","Unknown (edit
¥/etc¥/snmp¥/snmpd.conf)","public","notConfigUser","","","","",""]]
    [2] =>
{"Error":"¥u30e1¥u30f3¥u30c6¥u30ca¥u30f3¥u30b9¥u6a29¥u9650¥u304c¥u3042¥u308a¥u307e¥u305b
¥u3093¥u3002","Exception":"Generic error","StackTrace":"none"}
```

)
2020-11-06 13:32:52 Collect END ( Host name:ita-sample File name:RH_snmpd )

e.g.) Example of Log file output contents (Not target)
2020-11-05 16:55:31 [Process]The target device is not registered or is obsolete, so skip the registration and update process(Host name:ita-test)

## 5.3 BackYard contents

### 5.3.1 **Overview of the Parameter sheet registration process.**

(1) Acquire Collection interface information

(2) Acquire list over completed operations (with Normal end)
Collection target status: Complete

(3) Acquire the following information from the collectable operation no.
- Operation information
- Target host
- Target source file

(4) Inquire wether the target host is registered in the Device list or not
Registered: Collectable
Not registered: Not collectable

(5) Acquire the Menu ID of the target Parameter sheet from the source file and the Collection item value list.

(6) Create RESTAPI Parameter with the information gathered in Step 1-4.

Query the Menu ID for data and determine the RESTAPI Execution type.
Register: Unique operation and Host combination data is not registered.
Update: Unique operation and Host combination data is registered

(7) Register/Update the data using ITA Standard RESTAPI functions

(8) Update the status of the Collection status to the Operation No.

Keep in mind that the timing of the data registration to the Parameter sheet depends on the startup cycle of the Automatic process.

For more information regarding changing the startup cycle, please refer to “6.2 Maintenance”.

The access permission roles of the Registered/Updated records will inherit the access permission roles of the Collectable operation results.。

For more information regarding Target operation results, please refer to “Exastro-ITA_User_Instruction_Manual_Ansible-driver”.

# 6 Operation

Operation that uses this function includes: Inputs from users using browsers from client PCs and Operations done directly from the system operation/maintenance.

## 6.1 Maintenance

The following files are required to Start/Stop/Restart the Collect function process.

| Description | File name |
|---|---|
| Automatic Parameter registration<br>The operation is executed and will be registered to the parameter sheet based on the information registered in the setting item value list from the Operation results. | ky_std_synchronize-Collector.service |

The files are stored in 「/usr/lib/systemd/system」
The Start/Strop/Restart process methods are as following:
(Execute the commands with Root privileges)

① Start process

```
# systemctl start ky_std_synchronize-Collector.service
```

① Stop process

```
# systemctl stop ky_std_synchronize-Collector.service
```

② Restart process

```
# systemctl restart ky_std_synchronize-Collector.service
```

Replace each file name with the target file name and start/stop/restart.

## 6.2 Maintenance

① Change level to NORMAL
Rewrite the eigth row, “NORMAL, to “DEBUG”.
Log level settings file： <Install directory>/ita-root/confs/backyardconfs/ita_env

② Change level to DEBUG
Rewrite the eigth row, “DEBUG”, to “NORMAL”.
Log level settings file： <Install directory>/ita-root/confs/backyardconfs/ita_env

③ Change boot cycle.
Change the 5th parameter of ExecStart for each target file. (Unit: seconds)
Use the default value for boot cycles (except for exceptions).

```
ExecStart=/bin/sh ${ITA_ROOT_DIR}/backyards/common/ky_loopcall-php-procedure.sh
/bin/php /bin/php ${ITA_ROOT_DIR}/backyards/ansible_driver/ky_std_synchronize-
Collector.php ${ITA_ROOT_DIR}/logs/backyardlogs 10 ${ITA_LOG_LEVEL} > /dev/null
2>&1
```

Anything rewritten will take effect after the process is restarted.
Log file output destination：<Install directory>/ita-root/logs/backyardlogs

# 7 Appendix

## 7.1 References

Below are examples of IaCs (Playbook and Role)

1. Ansible Playbook Collection（Collect OS Settings）
   https://github.com/exastro-suite/playbook-collection-docs/blob/master/README.ja.md

2. Ansible config collecting and Parameter creating Playbook.

   makeYml_Ansible.yml

```
- name: make yaml file
  blockinfile:
    create: yes
    mode: 644
    insertbefore: EOF
    marker: ""
    dest: "{{ __parameter_dir__ }}/{{ inventory_hostname }}/Ansible_conf.yml"
    content: |
      ansible_architecture: {{ ansible_architecture }}
      ansible_bios_version: {{ ansible_bios_version }}
      ansible_default_ipv4__address: {{ ansible_default_ipv4.address }}
      ansible_default_ipv4__interface: {{ ansible_default_ipv4.interface }}
      ansible_default_ipv4__network: {{ ansible_default_ipv4.network }}
      ansible_distribution: {{ ansible_distribution }}
      ansible_distribution_file_path: {{ ansible_distribution_file_path }}
      ansible_distribution_file_variety: {{ ansible_distribution_file_variety }}
      ansible_distribution_major_version: {{ ansible_distribution_major_version }}
      ansible_distribution_release: {{ ansible_distribution_release }}
      ansible_distribution_version: {{ ansible_distribution_version }}
      ansible_machine: {{ ansible_machine }}
      ansible_memtotal_mb: {{ ansible_memtotal_mb }}
      ansible_nodename: {{ ansible_nodename }}
      ansible_os_family: {{ ansible_os_family }}
      ansible_pkg_mgr: {{ ansible_pkg_mgr }}
      ansible_processor_cores: {{ ansible_processor_cores }}
      ansible_processor_count: {{ ansible_processor_count }}
      ansible_processor_threads_per_core: {{ ansible_processor_threads_per_core }}
      ansible_processor_vcpus: {{ ansible_processor_vcpus }}
      ansible_product_name: {{ ansible_product_name }}
      ansible_product_serial: {{ ansible_product_serial }}
      ansible_product_uuid: {{ ansible_product_uuid }}
      ansible_product_version: {{ ansible_product_version }}
      ansible_python__executable: {{ ansible_python.executable }}
      ansible_python_version: {{ ansible_python_version }}
```

```
        ansible_service_mgr: {{ ansible_service_mgr }}
        ansible_php_config: php.ini
    delegate_to: localhost

  - name: get php config
    fetch:
      src: /etc/php.ini
      dest:   "{{ __parameters_file_dir__ }}/{{ inventory_hostname }}/"
      flat: yes
```

※ When you run makeYML_Ansible.yml and generate the Collectable source file (yaml), you need to enable gather_facts.
When editing the Movement list in Ansible Legacy, enter the following in the header section.
For details regarding Changing settings, pleaser refer to
“Exastro-ITA_User_Instruction_Manual_Ansible-driver”.

e.g) gather_facts Valid setting example.

```
  - hosts: all
    remote_user: "{{ __loginuser__ }}"
    gather_facts: yes
    become: yes
```