Exastro

PSSO Method Guidebook

~Optimizing Exastro System Construction/Operation~

Table of contents

- Introduction
- Overview Image.
- Automation Preparation
  - Step 1 : Central management of the Configuration info.
  - Step 2 : Actualize Automatic Execution.
  - Step 3 : Linking Central management and automatic execution.
- Implementing automated SI
  - Effects and Estimations
  - Post-Automation Process changes and results.
- Summary

Introduction

About this document

IT Engineers who are currently working in the field are struggling with inefficient system operation and construction. While the obvious solution is to make it more efficient, there are many who are wondering how to do it.

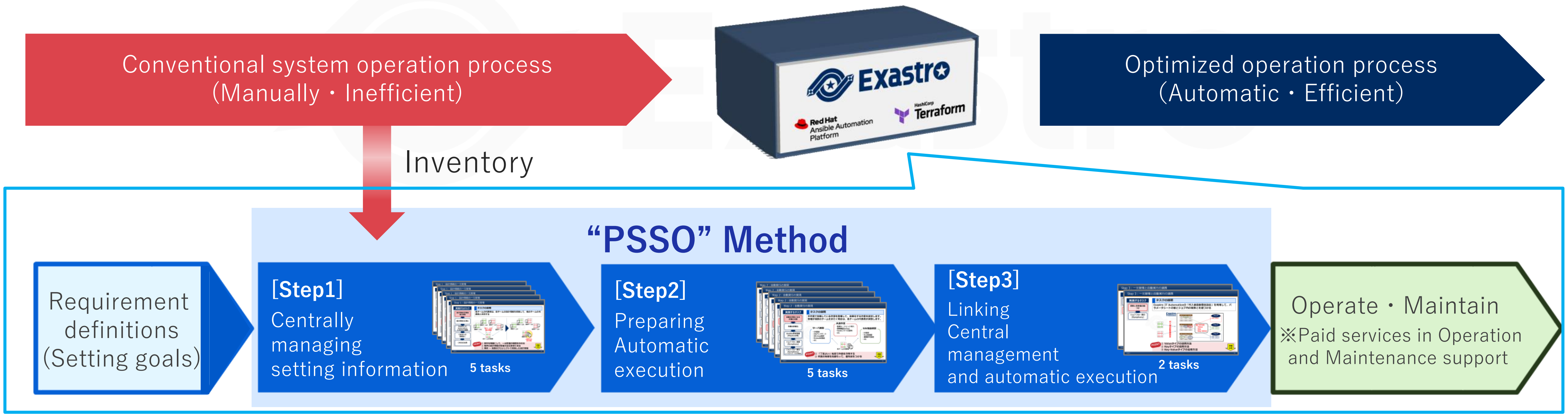
This document uses an on premise environment to show what obstacles to get rid off and what kind of preparation one must do in 3 simple steps **(AKA PSSO Method)**.

Step 1 : Central management of the Configuration info
Step 2 : Actualize Automatic Execution
Step 3 : Linking Central management and automatic execution.

In order to estimate the automation/efficiency rate, the process changes and results will be divided into phases.

What is the PSSO Method?

The PSSO (**P**rocedures for **S**treamlining **S**ystem **O**peration) method is a process of changing conventional "Manual system construction/operation" to "Automated system construction/operation" and solves problems often found during the Design, Preparation and Execution phases.



Conventional system operation process
(Manually・Inefficient)
Exastro
Red Hat Ansible Automation Platform
HashiCorp Terraform
Optimized operation process
(Automatic・Efficient)
Inventory
"PSSO" Method
Requirement definitions (Setting goals)
[Step1] Centrally managing setting information
5 tasks
[Step2] Preparing Automatic execution
5 tasks
[Step3] Linking Central management and automatic execution
2 tasks
Operate・Maintain
※Paid services in Operation and Maintenance support

Roles used in this document.

For the sake of convenience, we will explain the roles used in this document below.

Development/Construction team

- In this document, the team responsible for system construction will be called “Construction team”. Normally in a real project, this would also include someone responsible for business/affairs and infrastructure.

Operation team

- The team responsible for operating running systems is called “Operation team”.

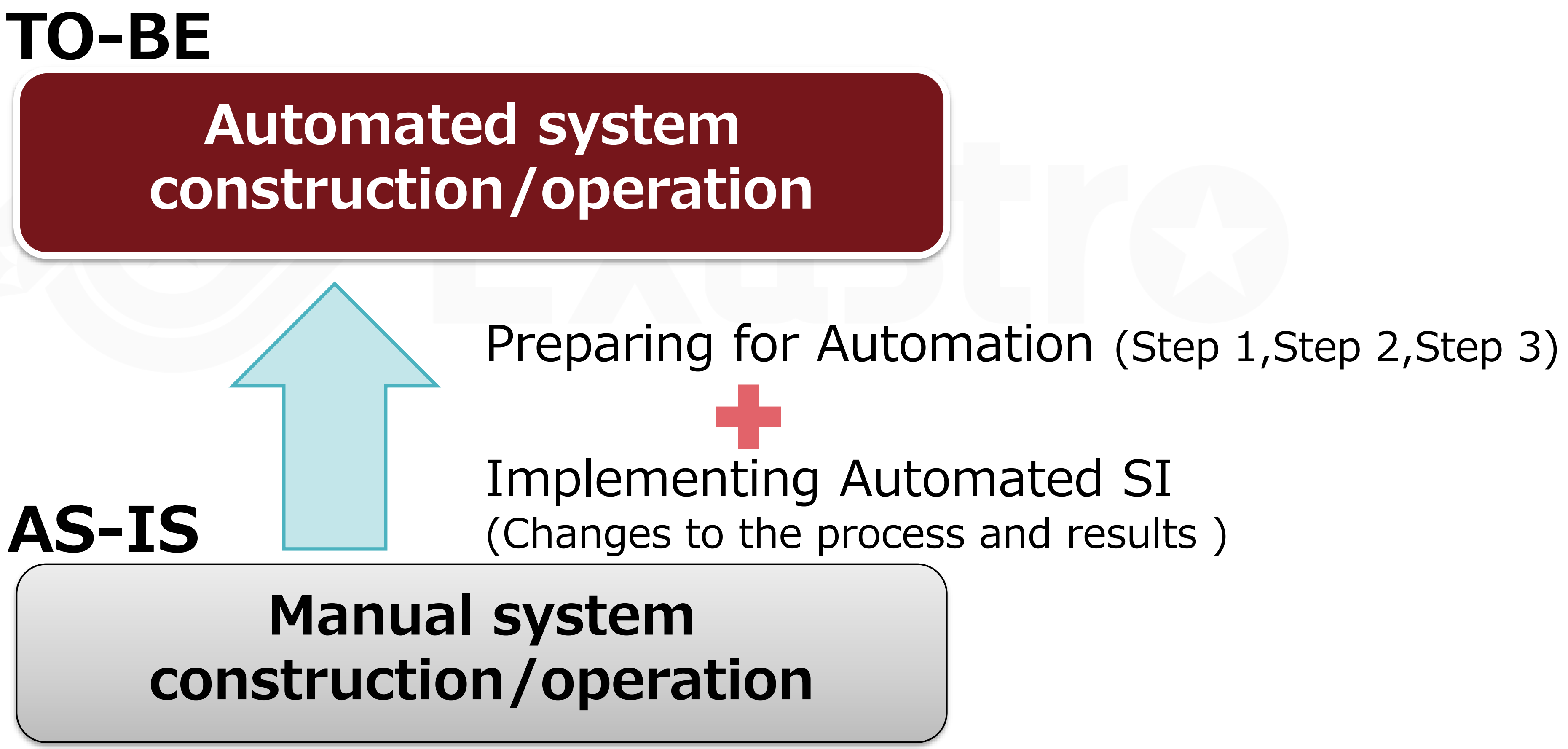
Team leader

- Representatives from each team who shares information and coordinates the team.

Overview image

AS-IS and TO BE in Automation.

By following step 1-3, we can automate system operation/construction. Additionally, by changing the process, we can improve the efficiency of the automation.


TO-BE
Automated system construction/operation
Preparing for Automation (Step 1,Step 2,Step 3)
Implementing Automated SI
(Changes to the process and results)
AS-IS
Manual system construction/operation

The “pain” of IT Engineers

The “pain” of IT Engineers that works with Constructing/Operating systems

- ☑ Delays and errors occurs when communicating between teams.
- ☑ Double managing data and proprietary wording leads to errors in the design
- ☑ Multiple development leads to complications with managing design documents (forms)
- ☑ As a result, we don’t know what information is the newest.

- ☑ Work orders between teams are complex. Each time a time chart is created, it gets discarded.
- ☑ Every operation’s Manual is discarded after its created/reviewed.
- ☑ Configurations are embedded in each procedure, and the number of patterns increases each time a new model/OS is added (barrier to multi-vendor support)

- ☑ Since the operations are done manually, the production time is inconsistent.
  ⇒People often have to wait before they can continue.
- ☑ Since most of the operations are done manually, human error is inevitable.

The “Pain” of IT Engineers

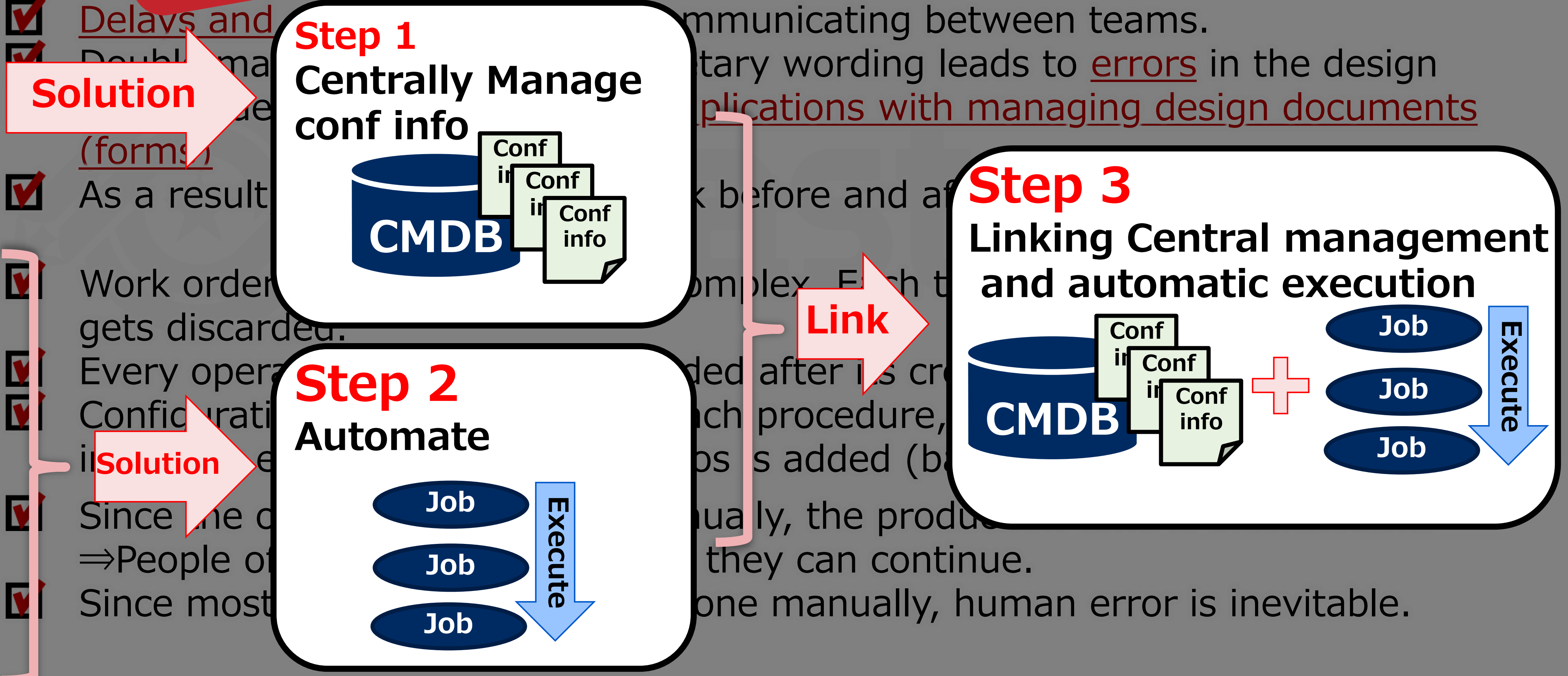
Said problems can be solved in 3 Steps

POINT **Step 1 is the most important step in the PSSO Method**

- ☑ Delays and … mmunicating between teams.
- ☑ Doubl… ma… tary wording leads to errors in the design
- ☑ …de… plications with managing design documents (forms)
- ☑ As a result … k before and af…

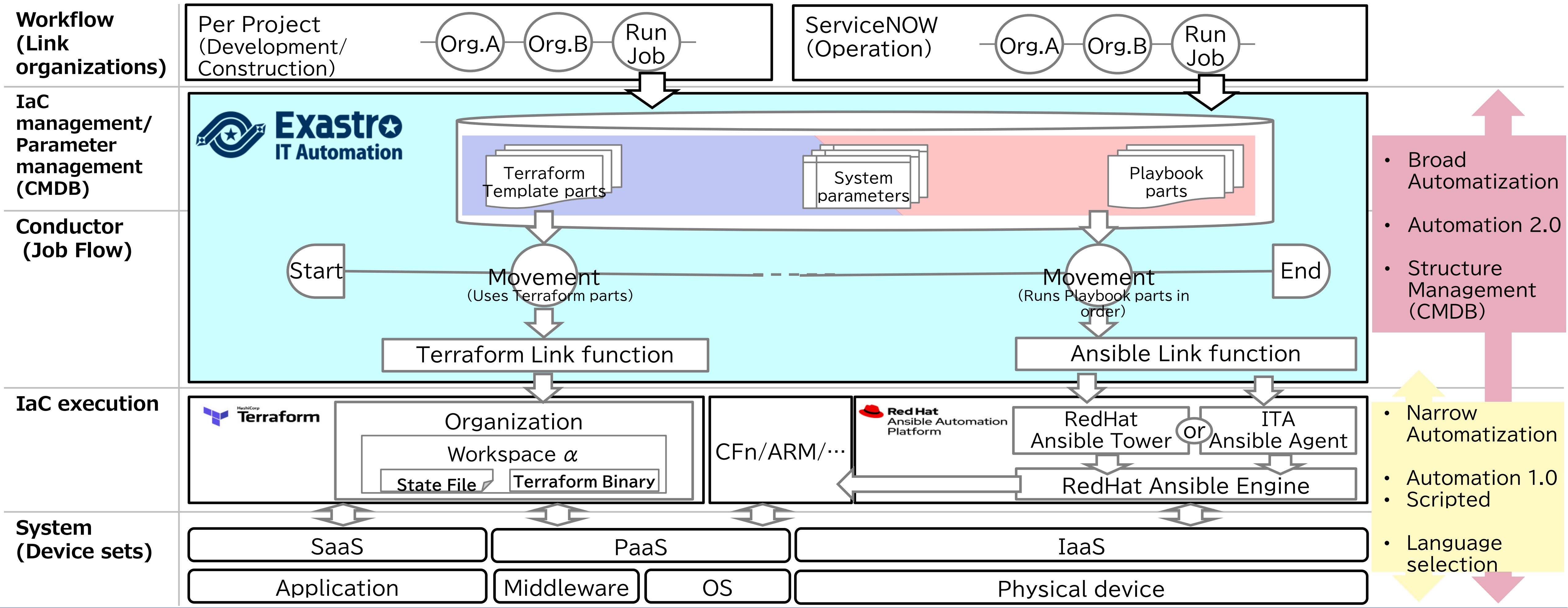
- ☑ Work order … omplex. Each t… gets discarded.
- ☑ Every opera… ded after its cr…
- ☑ Configurati… ch procedure, i… e… ps is added (ba…

- ☑ Since the … ually, the produ…
  ⇒People of … they can continue.
- ☑ Since most … one manually, human error is inevitable.



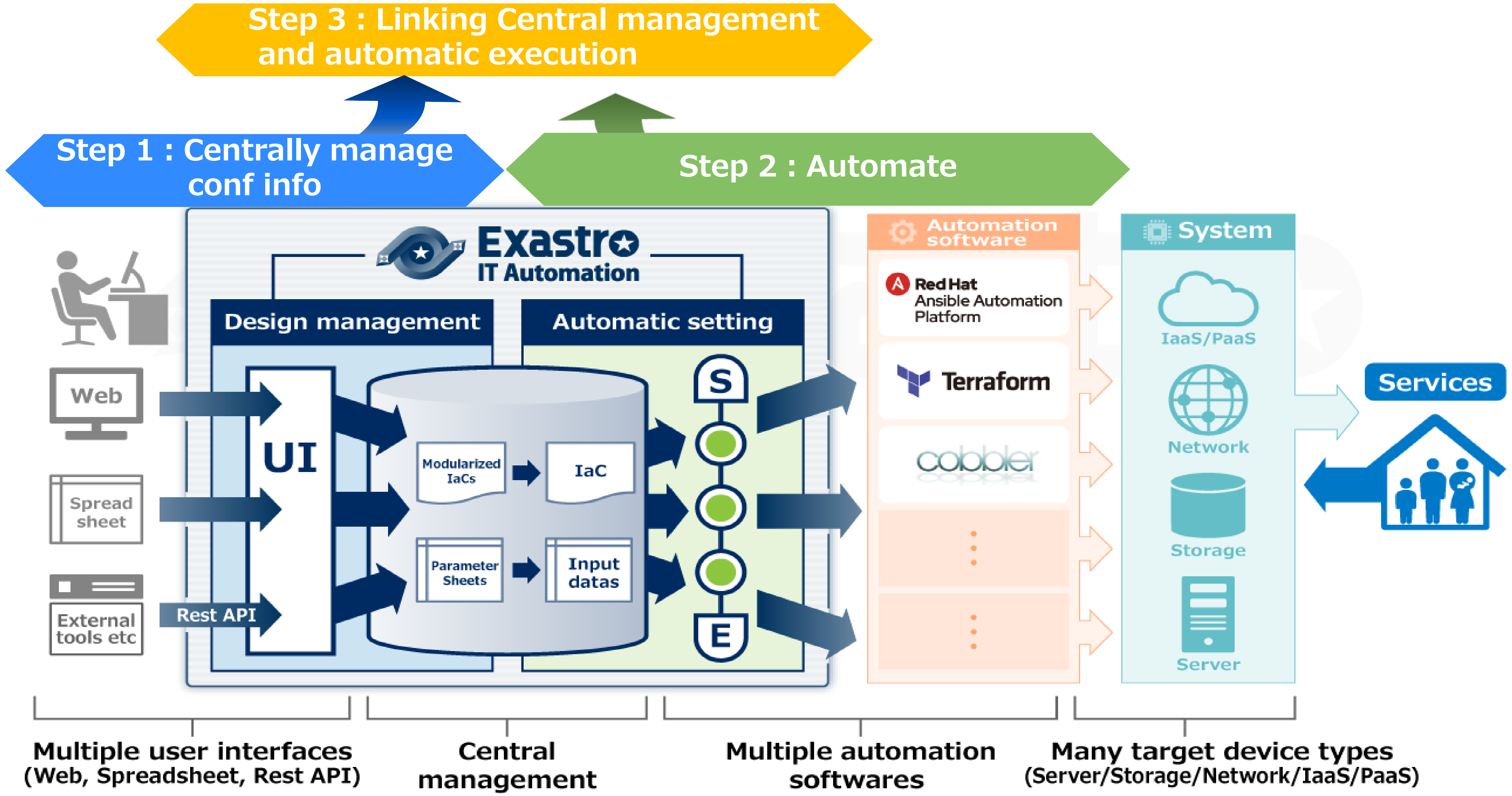
Relationship between the Automatization scope and the different steps

Step 2 only ⇒ "Narrow Automatization"
Step 1～Step 3 ⇒ "Broad Automatization"



Overview over Exastro IT Automation

Exastro IT Automation supports the 3 step solution

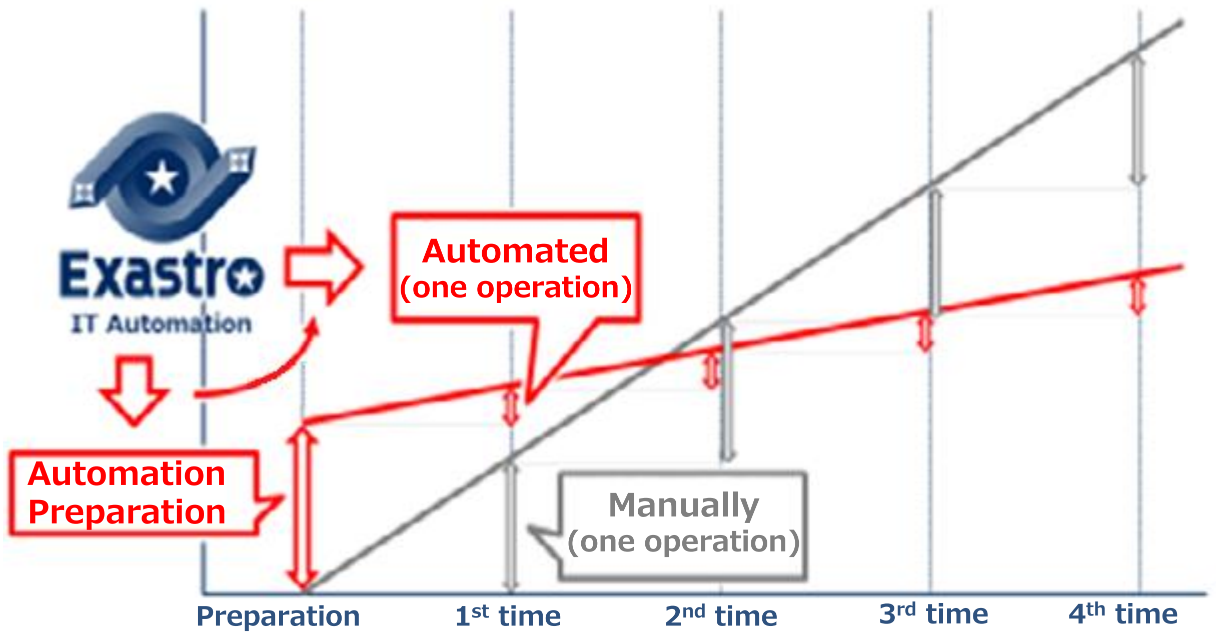


Step 3 : Linking Central management and automatic execution
Step 1 : Centrally manage conf info
Step 2 : Automate
Exastro IT Automation
Design management
Automatic setting
Web
Spread sheet
External tools etc
UI
Rest API
Modularized IaCs
IaC
Parameter Sheets
Input datas
S
E
Automation software
Red Hat Ansible Automation Platform
Terraform
cobbler
System
IaaS/PaaS
Network
Storage
Server
Services
Multiple user interfaces (Web, Spreadsheet, Rest API)
Central management
Multiple automation softwares
Many target device types (Server/Storage/Network/IaaS/PaaS)

Automation changes QCD and Tasks/results.

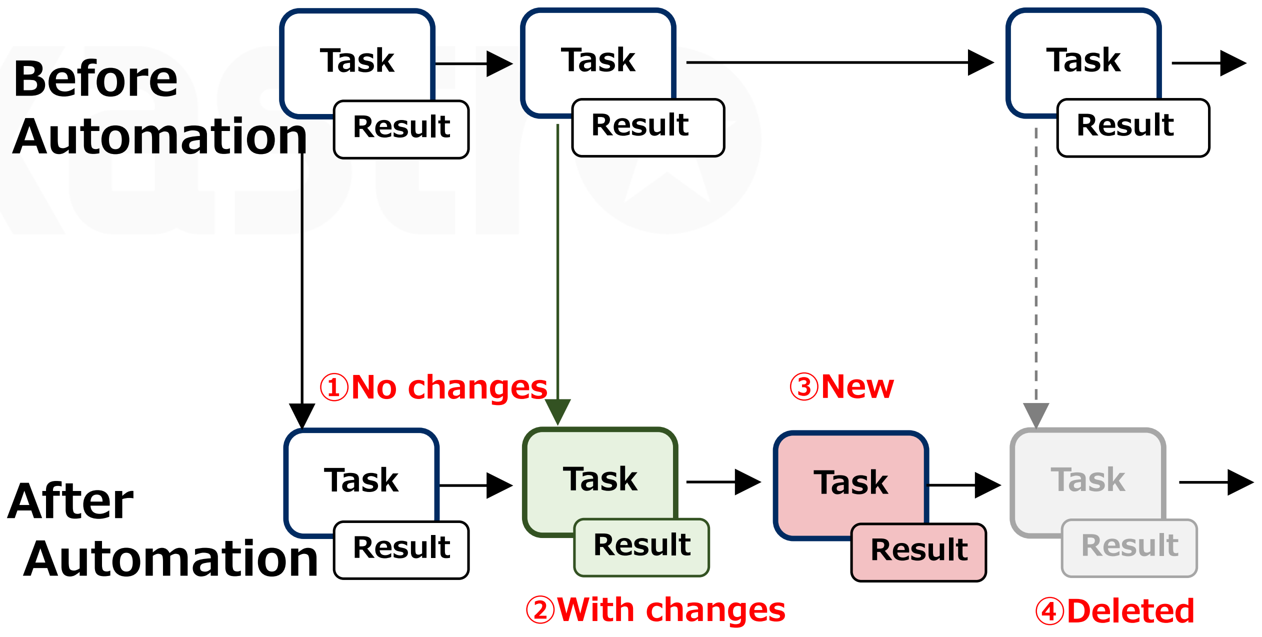
QCD (Quality · Cost · Delivery)

Manual labor→ QCD Reform from Automation.



Tasks and Results

Tasks and Result changes can be divided in these 4 groups → 1.No changes 2.With changes 3.New 4,Deleted



Automation Preparation

Step 1 : Central management of the Configuration info.

Step 2 : Actualize Automatic Execution.

Step 3 : Linking Central management and automatic execution.

Exastro

Automation Preparation

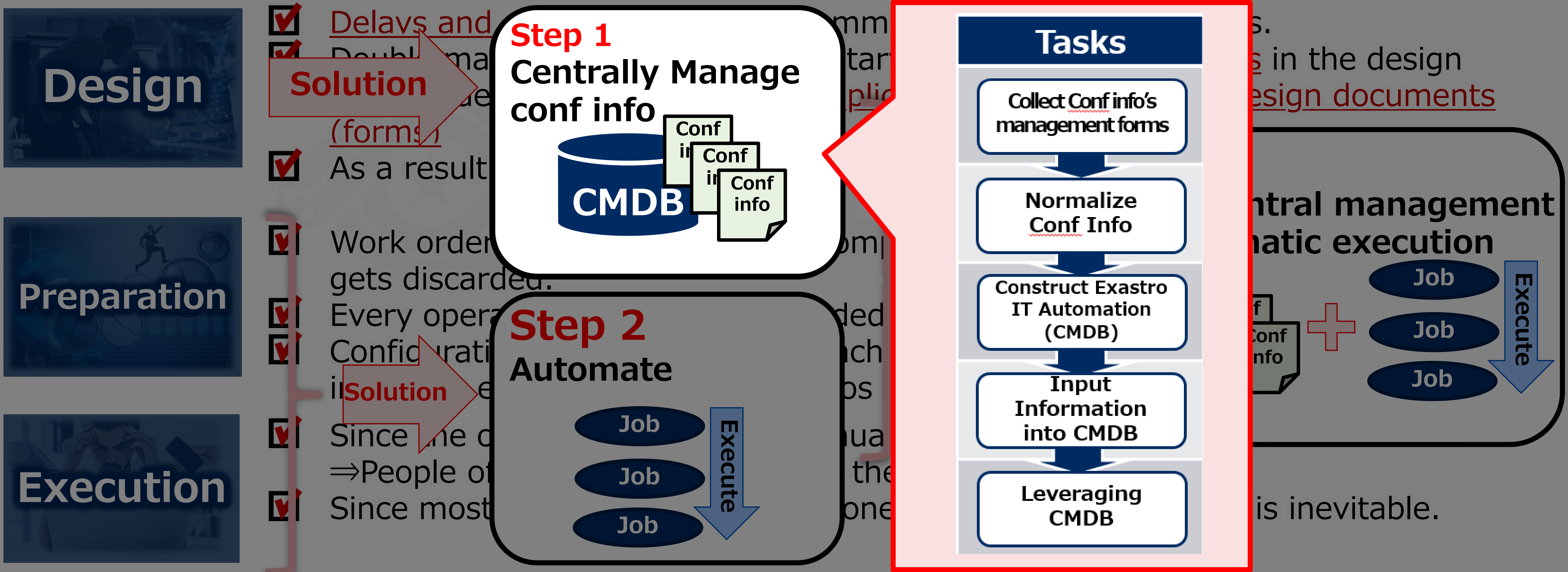
Step 1 : Central management of the Configuration info.

Step 2 : Actualize Automatic Execution.

Step 3 : Linking Central management and automatic execution.

Step 1 : Central management of the Configuration info

The next slides explains the 5 tasks in Step 1.



Tasks

1. Collect Conf info's management forms
2. Normalize Conf Info
3. Construct Exastro IT Automation (CMDB)
4. Input Information into CMDB
5. Leveraging CMDB

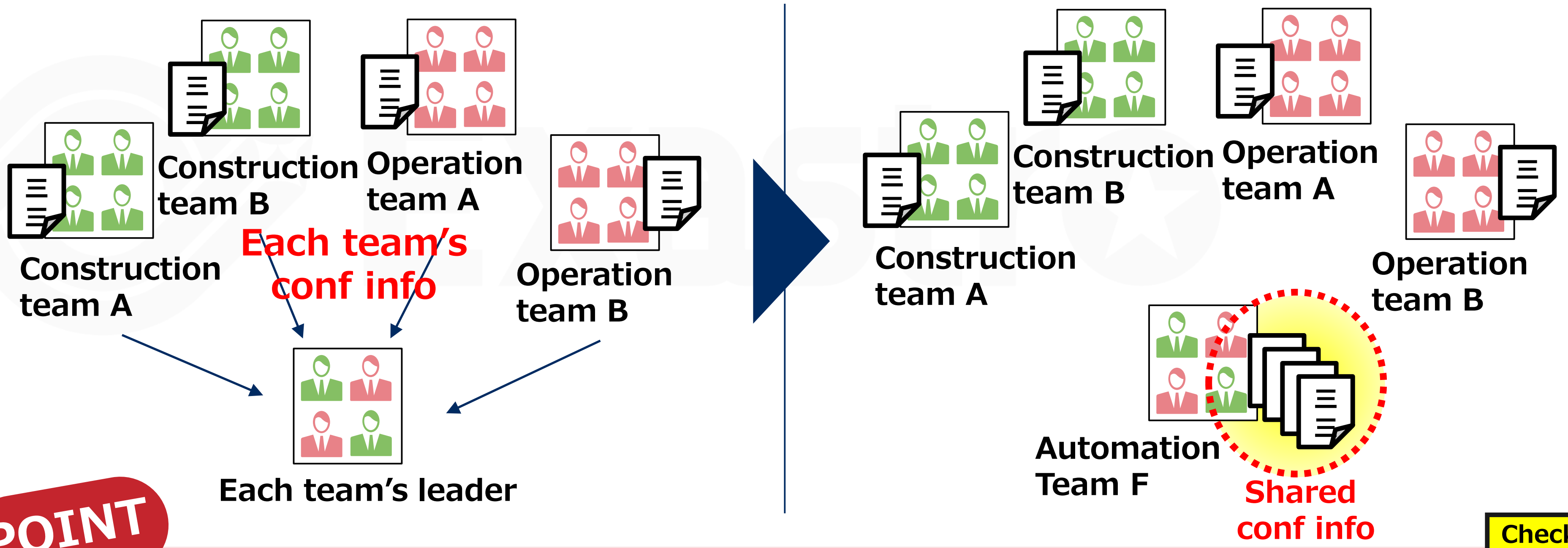
Step 1 : Central management of the Configuration info

Tasks

- **Collect Conf info's management forms**
- Normalize Conf Info
- Construct Exastro IT Automation (CMDB)
- Input Information into CMDB
- Leveraging CMDB

Task Explanation

Each team leader collects the conf info from their own teams and share it with each other.



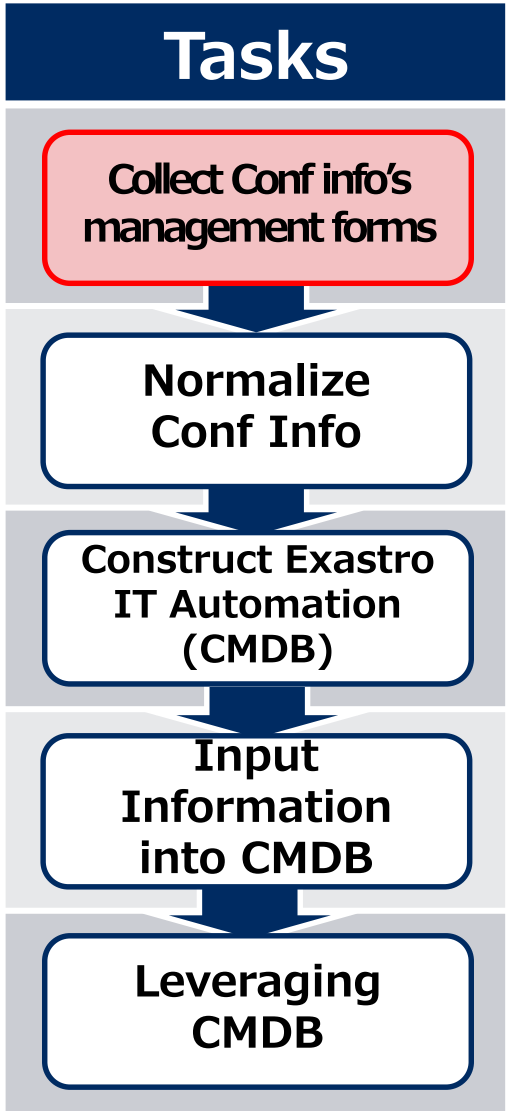
POINT

① Clarify the purpose and decide the scope of the management
② There are several ways to manage existing conf info
③ Example) conf info collected from an actual project.

Check next page

Step 1 : Central management of the Configuration info

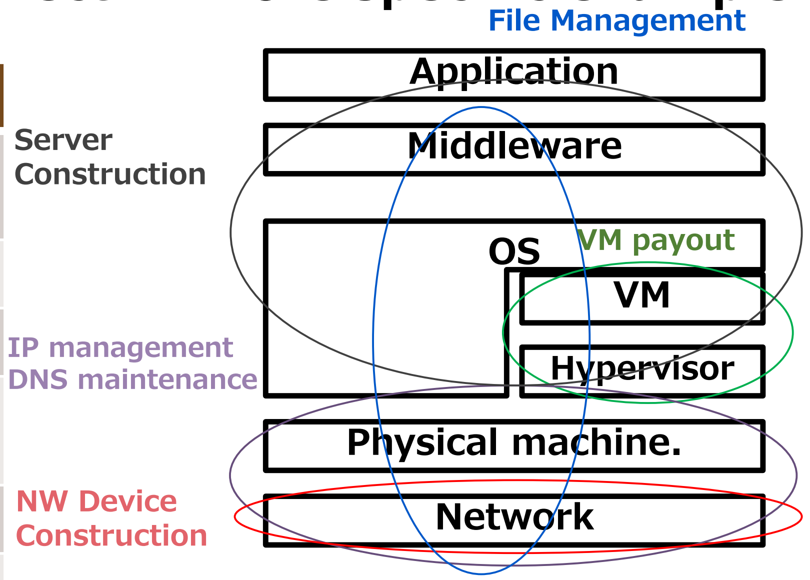
Tasks


Collect Conf info's management forms
Normalize Conf Info
Construct Exastro IT Automation (CMDB)
Input Information into CMDB
Leveraging CMDB

POINT ① **Clarify the purpose and decide the scope of the management**

First, one should clarify the goal. After that, we can decide the scope of the conf information we want to collect. A more specific example can be found below。

Goals often used	Scope of information
1) IP Address Management	IP, Segments, Etc.
2) Assets Management	Serial Number, License, etc.
3) Server construction	IP, Host name, etc.
4) NW device construction	Interface Numbers, VLAN, etc.
5) VM Payout	Hypervisor, VM name, etc.
6) DNS maintaining	DNS server, domain name, etc.



Problems such as collecting too much or unnecessary information may occur if there are no clear goals.
If there are multiple goals, we recommend to number them by priority and create the CMDBs in order.

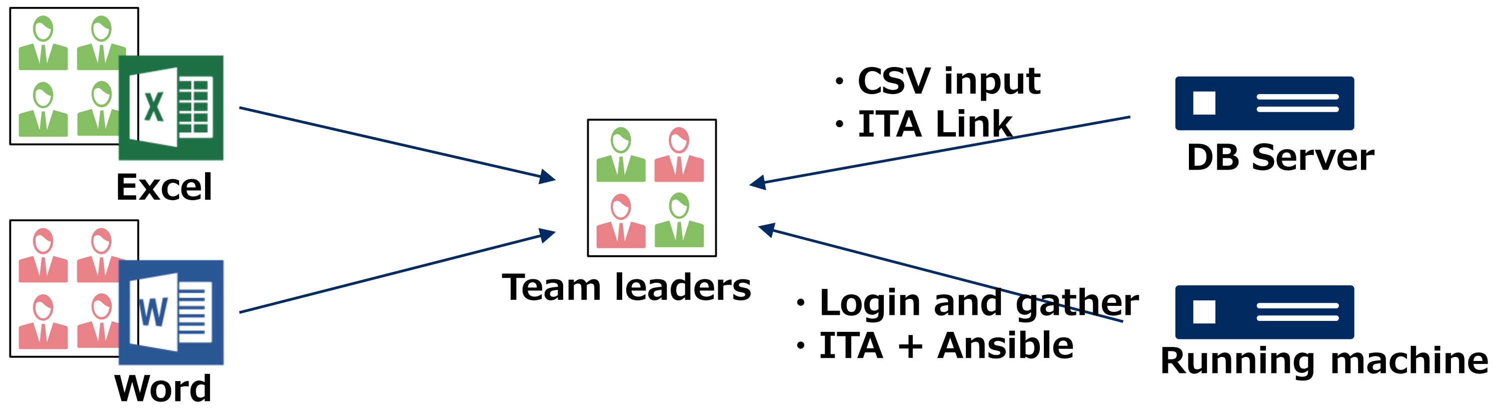
Step 1 : Central management of the Configuration info

Tasks

- Collect Conf info's management forms
- Normalize Conf Info
- Construct Exastro IT Automation (CMDB)
- Input Information into CMDB
- Leveraging CMDB

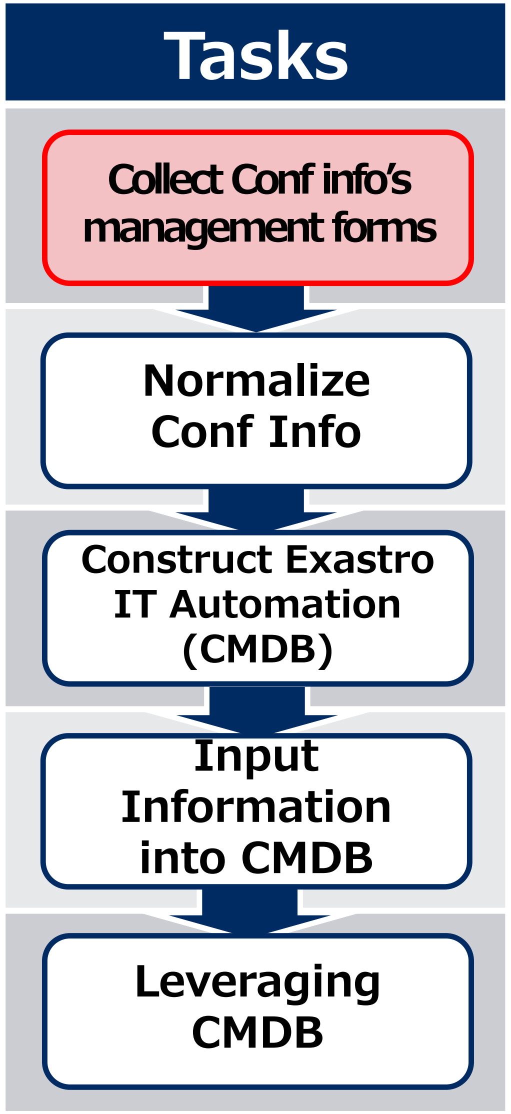
POINT ② There are several ways to manage existing conf info

Many projects uses Excel or Word formats to manage conf info, so let's start with collecting those files. If you are storing conf info in databases, you might consider dumping it in CSV Format or to link the database directly with Exastro IT Automation.



Depending on the project, users might have to gather information straight from a running machine (such as a VM) instead of the conf info documents. In that case, we can use Exastro ITA and Ansible to easily collect data from the machines.

Step 1 : Central management of the Configuration info


Tasks
Collect Conf info's management forms
Normalize Conf Info
Construct Exastro IT Automation (CMDB)
Input Information into CMDB
Leveraging CMDB

POINT

③ Case ～ Collecting Conf info from a real project.

Here is an example of how Construction management of servers and network devices can be achieved. In this case, the following conf info was shared among the team leaders in order to easily identify the scope of the outage impact of the service.

Team	Collected Conf info
Server G	・Server list ・Software installed on the server list
Network G	・IP address list ・Network device list ・Network route list
Storage G	・Path list ・Storage disk list
Operation monitoring G	・Message list
Business G	・Components list ・Server components list ・Communication conditions list

For more details regarding this case, please refer to the URL below.
https://exastro-suite.github.io/it-automation-docs/case.html

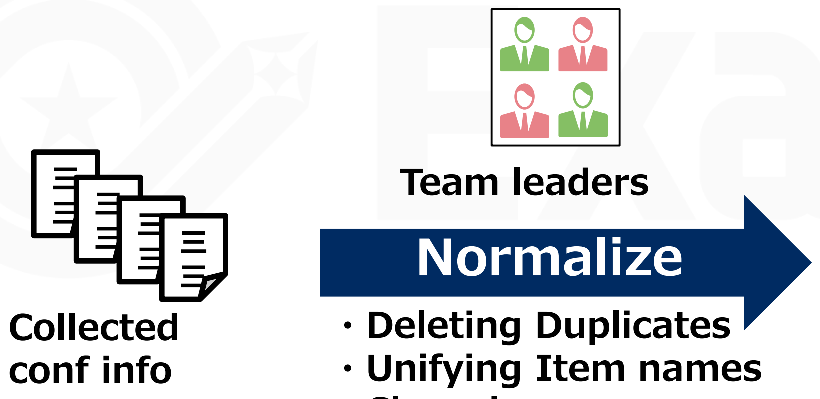
Step 1 : Central management of the Configuration info

Tasks

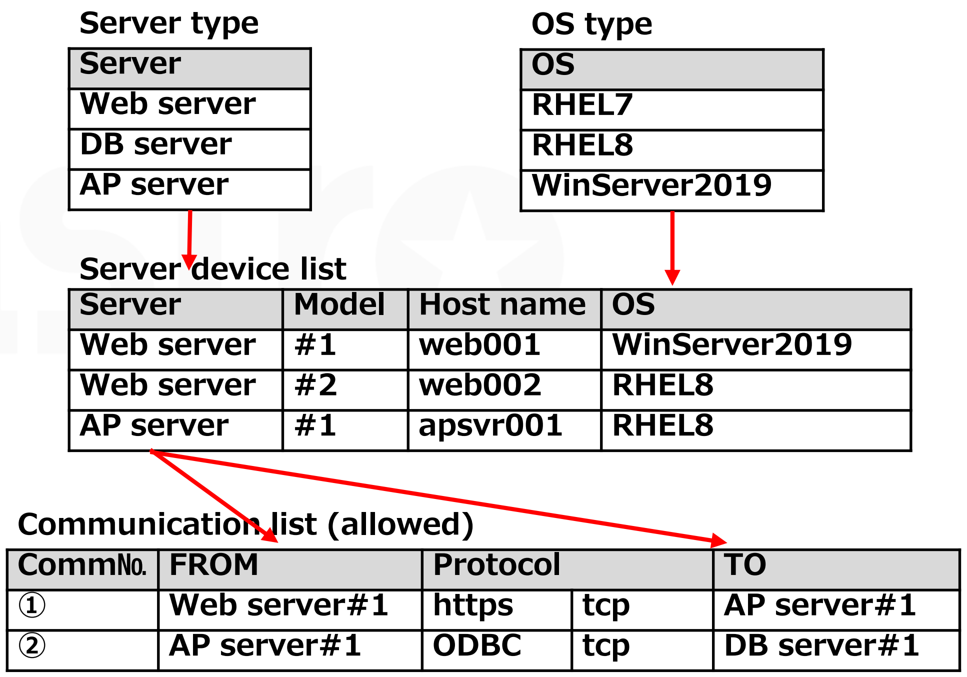
- Collect Conf info's management forms
- **Normalize Conf Info**
- Construct Exastro IT Automation (CMDB)
- Input Information into CMDB
- Leveraging CMDB

Task explanation

The team leaders normalizes the collected conf info in a table format by eliminating duplicates, unifying names and breaking up redundant info.



・Deleting Duplicates
・Unifying Item names
・Cleansing
・Etc.



Server type

Server
Web server
DB server
AP server

OS type

OS
RHEL7
RHEL8
WinServer2019

Server device list

Server	Model	Host name	OS
Web server	#1	web001	WinServer2019
Web server	#2	web002	RHEL8
AP server	#1	apsvr001	RHEL8

Communication list (allowed)

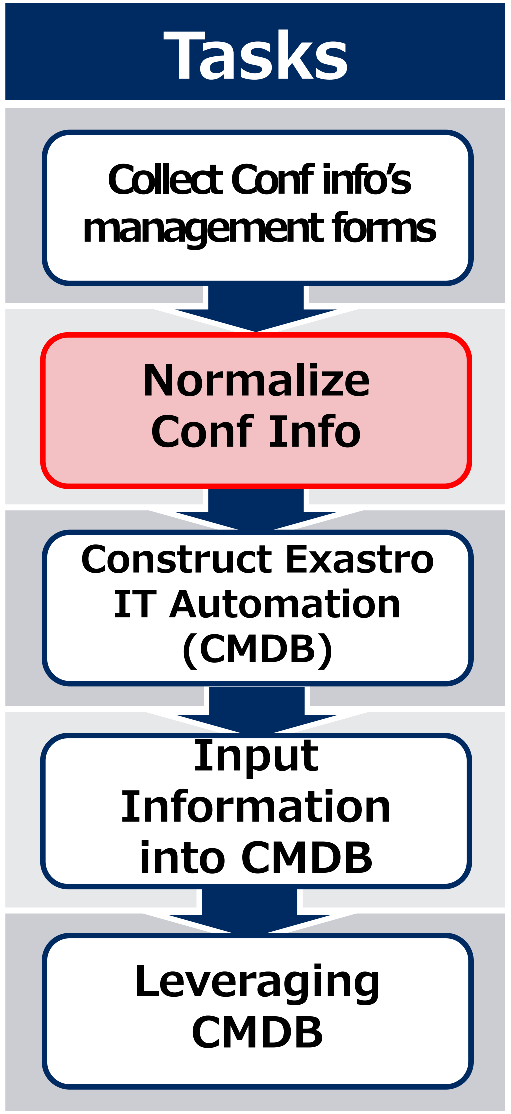
CommNo.	FROM	Protocol		TO
①	Web server#1	https	tcp	AP server#1
②	AP server#1	ODBC	tcp	DB server#1

POINT

① Sort the conf info
② Organize the conf info items (Columns)

Check next page

Step 1 : Central management of the Configuration info


Tasks
Collect Conf info's management forms
Normalize Conf Info
Construct Exastro IT Automation (CMDB)
Input Information into CMDB
Leveraging CMDB

POINT ① Sort the conf info

Each team's collected conf info is sorted according to the following

① If the info is enclosed to single teams or if it is shared with other teams.

If there is info linked with other teams, separate it from other info. By doing so, we can share the info with each others.

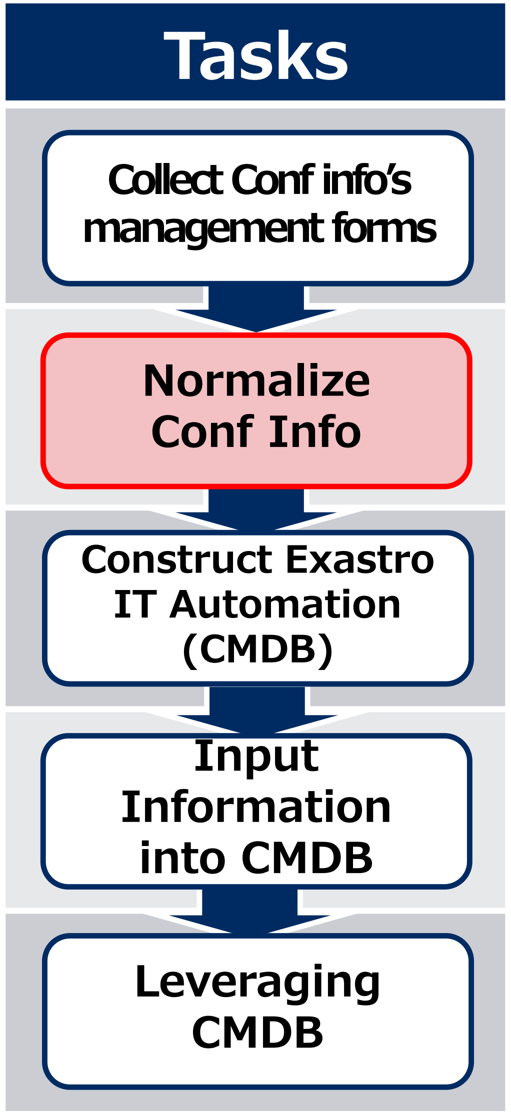
② If we're making the user select info from a pull-down menu in Exastro ITA.

We divide the info into two categories when registering conf info. Info selectable from pull-down menus and info that can be entered manually. Info selected from pull-down menus will have their values registered as "Master".

③ The relationship of the conf information.

We must decide the relationship (dependency) of the conf info. This is important, as it directly affects the order in which we create and register conf info. For example, in order to create a "server list" , we first have to create and register "OS types".

Step 1 : Central management of the Configuration info


Tasks
Collect Conf info's management forms
Normalize Conf Info
Construct Exastro IT Automation (CMDB)
Input Information into CMDB
Leveraging CMDB

POINT **② Organize the conf info items (Columns)**

Eventually, the conf info is collected in a table format. Therefore, it is necessary to organize what the "column" in the table should be according to the points below.

① Unification of the settings info item names (table column names).

Different teams often have different names for the same information. For example, the server team might call "IP Address" for just "IP", while the network team might call it for "ip_addr". In this case, we need to have the teams use the same name so the information can be counted as shared conf info.

② Grouping the settings info.

In many cases, settings info becomes more readable if it is grouped up. To give an example, by grouping "IP Address"# and "Port Number" into "Connection Information", we can improve both the readability and maintainability.

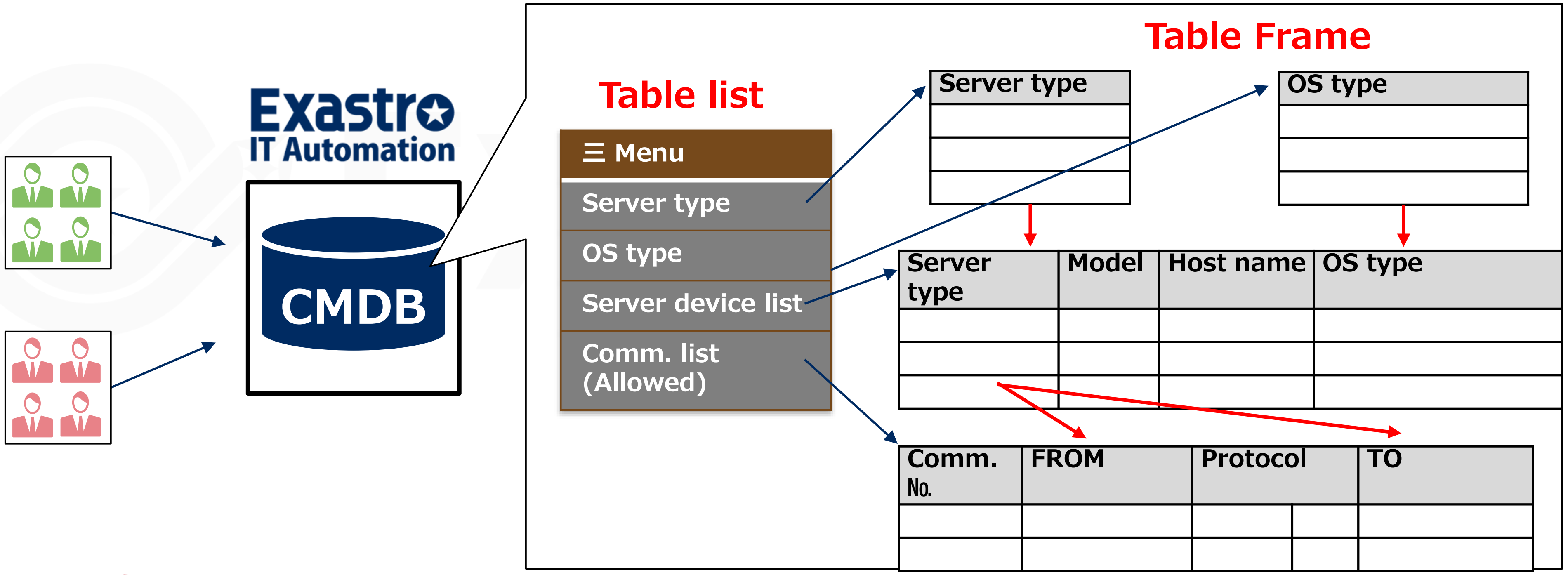
Step 1 : Central management of the Configuration info

Tasks

- Collect Conf info's management forms
- Normalize Conf Info
- Construct Exastro IT Automation (CMDB)
- Input Information into CMDB
- Leveraging CMDB

Task explanation.

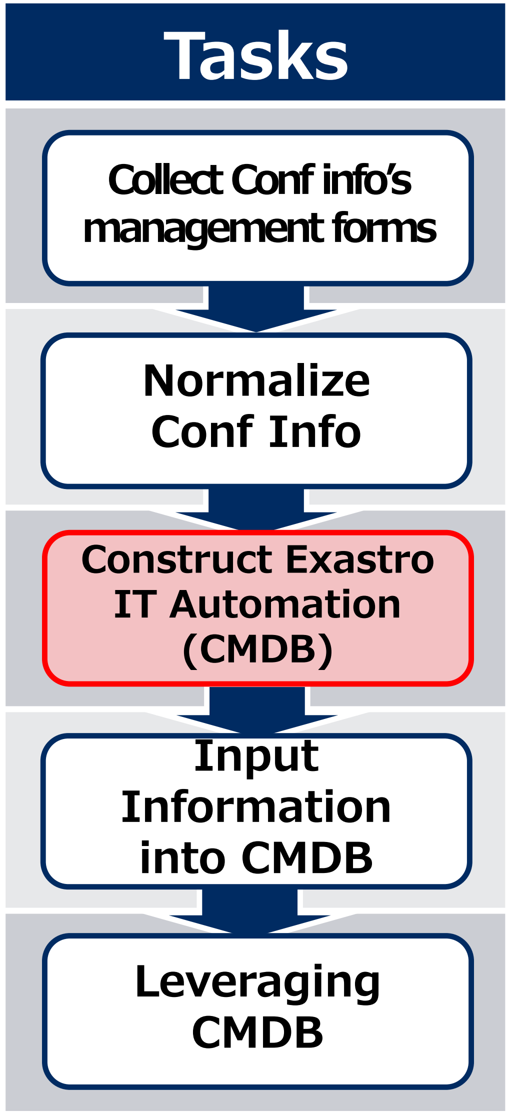
Based on the normalized conf info, create a "table list" and a "table frame" to store the conf info in the CMDB in Exastro IT Automation.



POINT ① **Put restrictions on the columns to prevent input errors in the design values.**

Check next page

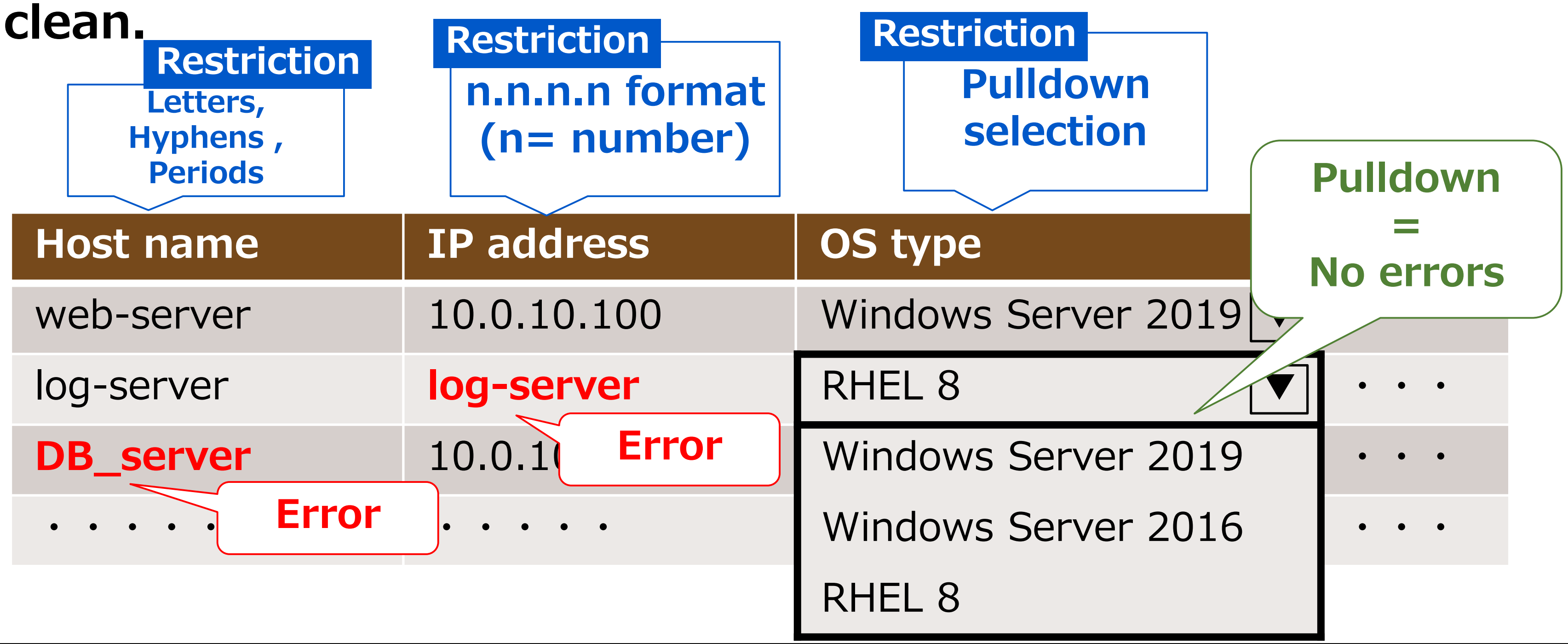
Step 1 : Central management of the Configuration info


Tasks
Collect Conf info's management forms
Normalize Conf Info
Construct Exastro IT Automation (CMDB)
Input Information into CMDB
Leveraging CMDB

POINT ① **Put restrictions on the columns to prevent input errors in the design values**

Keeping the CMDB clean is impossible there are spelling/input errors when registering design values.

By setting restrictions to the table columns in Exastro IT Automation, it becomes easier if there any spelling/input errors when inputting new design values. As a result, the CMDB can be kept clean.



Host name	IP address	OS type	
web-server	10.0.10.100	Windows Server 2019	
log-server	log-server	RHEL 8 ▼	・・・
DB_server	10.0.1		・・・
・・・・・	・・・・・		・・・

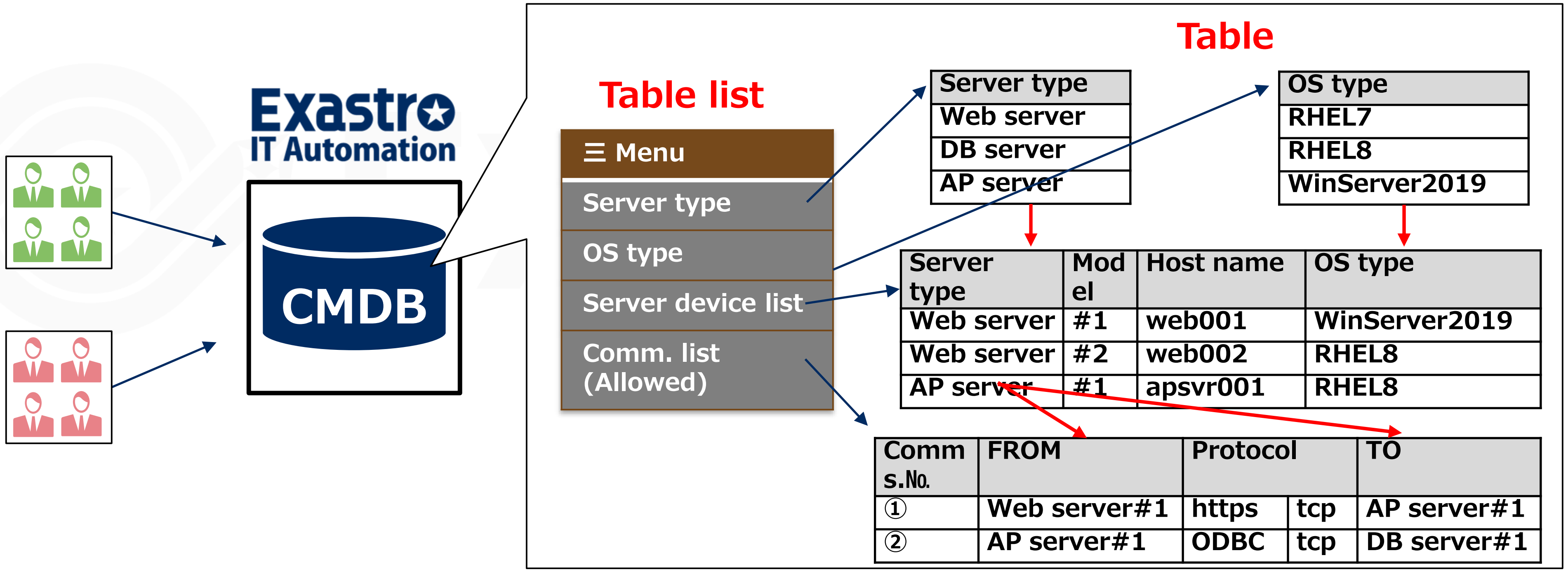
Step 1 : Central management of the Configuration info

Tasks

- Collect Conf info's management forms
- Normalize Conf Info
- Construct Exastro IT Automation (CMDB)
- **Input Information into CMDB**
- Leveraging CMDB

Task explanation

Every team registers the conf info to the CMDB



Server type
Web server
DB server
AP server

OS type
RHEL7
RHEL8
WinServer2019

Server type	Model	Host name	OS type
Web server	#1	web001	WinServer2019
Web server	#2	web002	RHEL8
AP server	#1	apsvr001	RHEL8

Comms.No.	FROM	Protocol		TO
①	Web server#1	https	tcp	AP server#1
②	AP server#1	ODBC	tcp	DB server#1

POINT ① Use Excel to register in batches.

Check next page

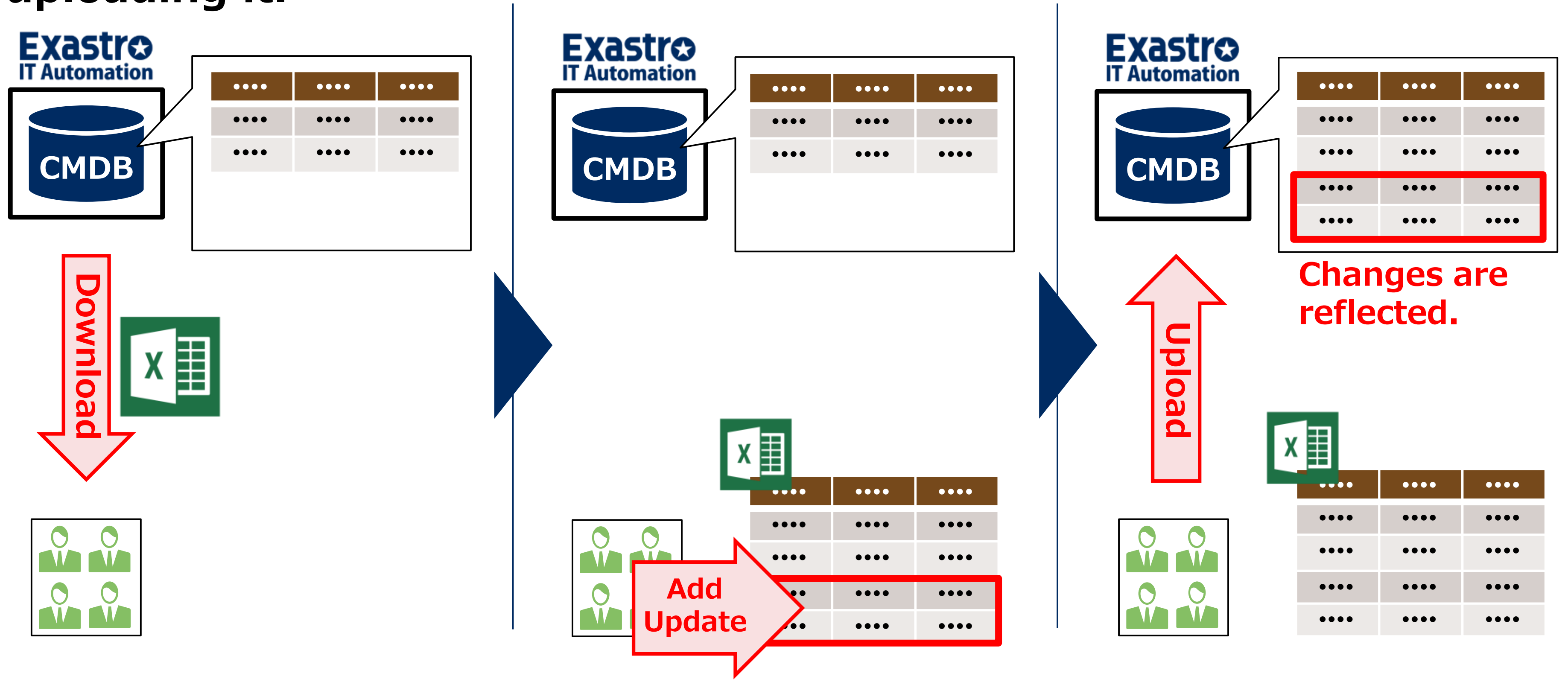
Step 1 : Central management of the Configuration info

Tasks

- Collect Conf info's management forms
- Normalize Conf Info
- Construct Exastro IT Automation (CMDB)
- **Input Information into CMDB**
- Leveraging CMDB

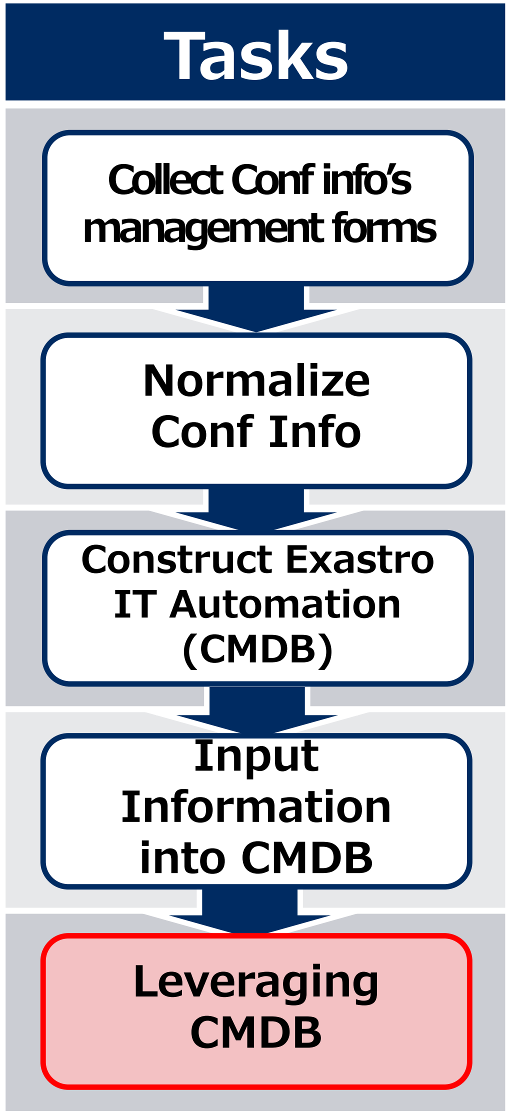
POINT ① **Use Excel to register in batches.**

The tables in Exastro IT Automation can be downloaded in Excel format. We can register conf info more efficiently by adding/updating the information directly to the Excel file and then uploading it.



Step 1 : Central management of the Configuration info

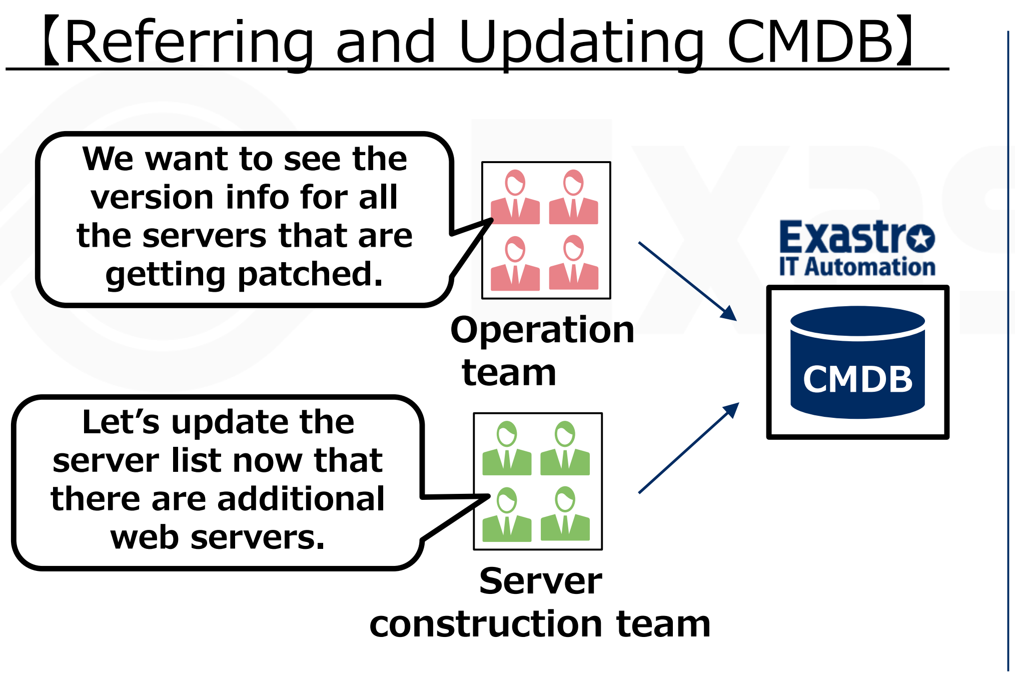
Tasks



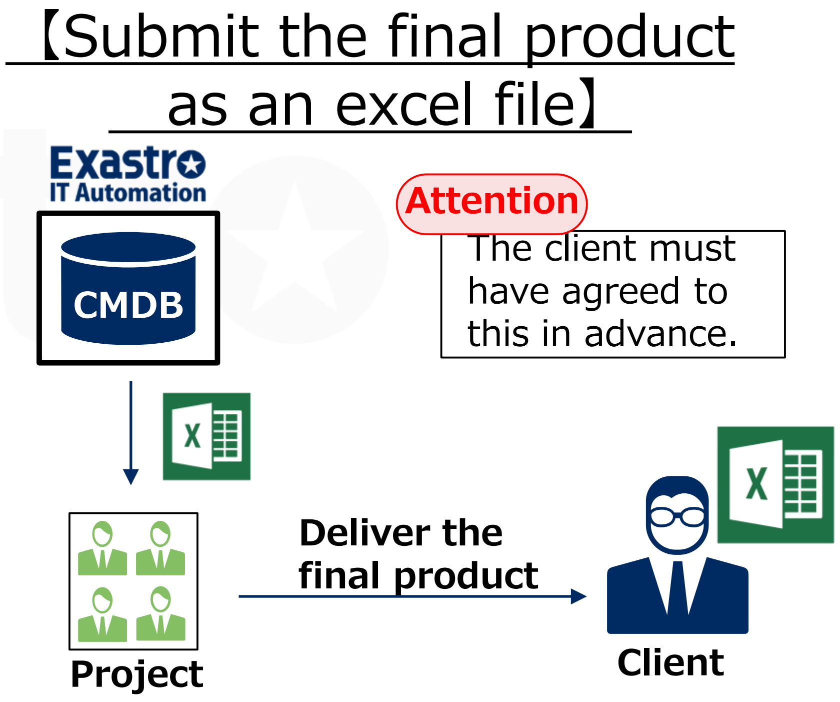
Collect Conf info's management forms
Normalize Conf Info
Construct Exastro IT Automation (CMDB)
Input Information into CMDB
Leveraging CMDB

Task explanation

Refer and update the conf info to suit the final goal.
Additionally, it is possible to store the setting values by downloading it as an Excel file.



【Referring and Updating CMDB】
We want to see the version info for all the servers that are getting patched.
Operation team
Let's update the server list now that there are additional web servers.
Server construction team
Exastro IT Automation
CMDB



【Submit the final product as an excel file】
Exastro IT Automation
CMDB
Attention
The client must have agreed to this in advance.
Project
Deliver the final product
Client

POINT ① Case~ Investigating the scope of service outage impacts.

Check next page

Step 1 : Central management of the Configuration info

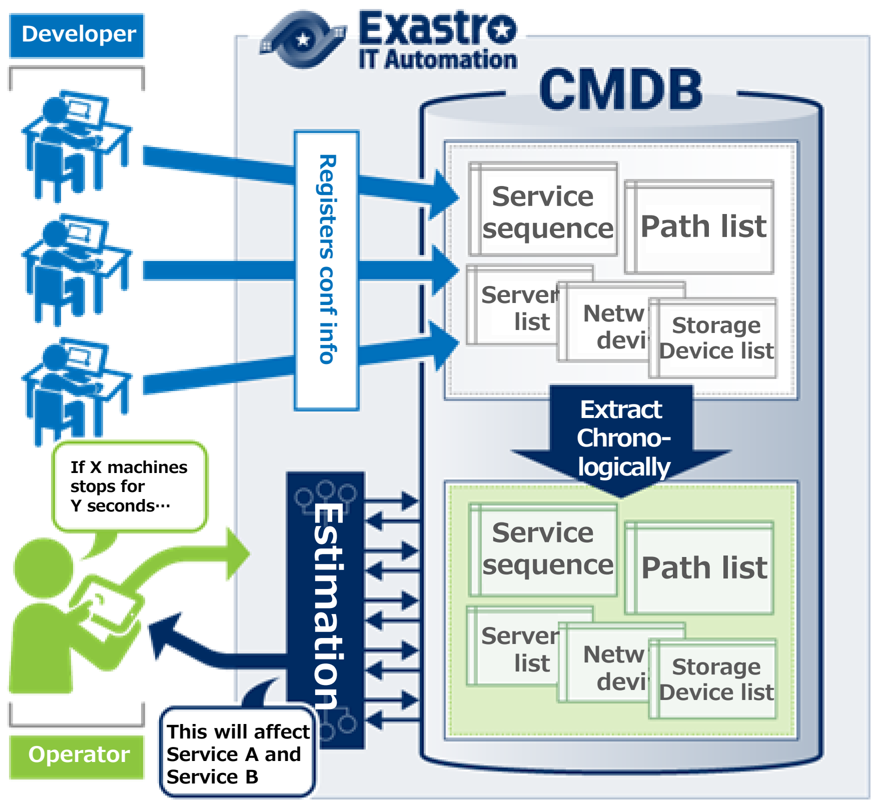
Tasks

- Collect Conf info's management forms
- Normalize Conf Info
- Construct Exastro IT Automation (CMDB)
- Input Information into CMDB
- **Leveraging CMDB**

POINT ① **Case~ Investigating the scope of service outage impacts.**

Here, we will show an example of using CMDB to investigate the impact of a service outage.

Problem	Large-scale carrier systems require a lot of man-hours to investigate service impacts of both expected and unexpected equipment outages.
	Construct CMDB
Solution	By managing the configuration of the system, it is possible to automatically predict the impact of equipment outages.
Effect	Don't have to pay 800 000 Yen per investigation. The annual cost was reduced by about 94 mil. Yen. (checked 120 times)



For more details regarding this case, please refer to the URL below.
https://exastro-suite.github.io/it-automation-docs/case.html#case003

Automation Preparation

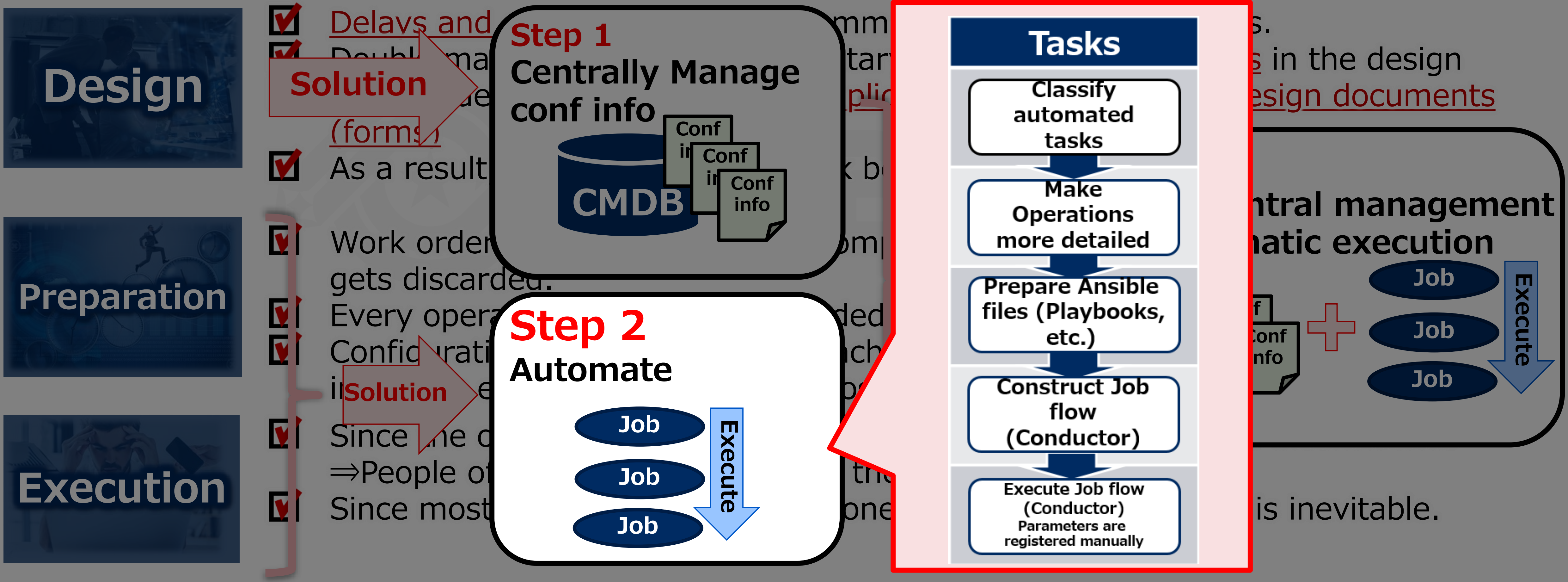
Step 1 : Central management of the Configuration info.

Step 2 : Actualize Automatic Execution.

Step 3 : Linking Central management and automatic execution.

Step 2 : Actualize Automatic Execution

The next slides explains the 5 tasks in Step 2.



Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor) Parameters are registered manually

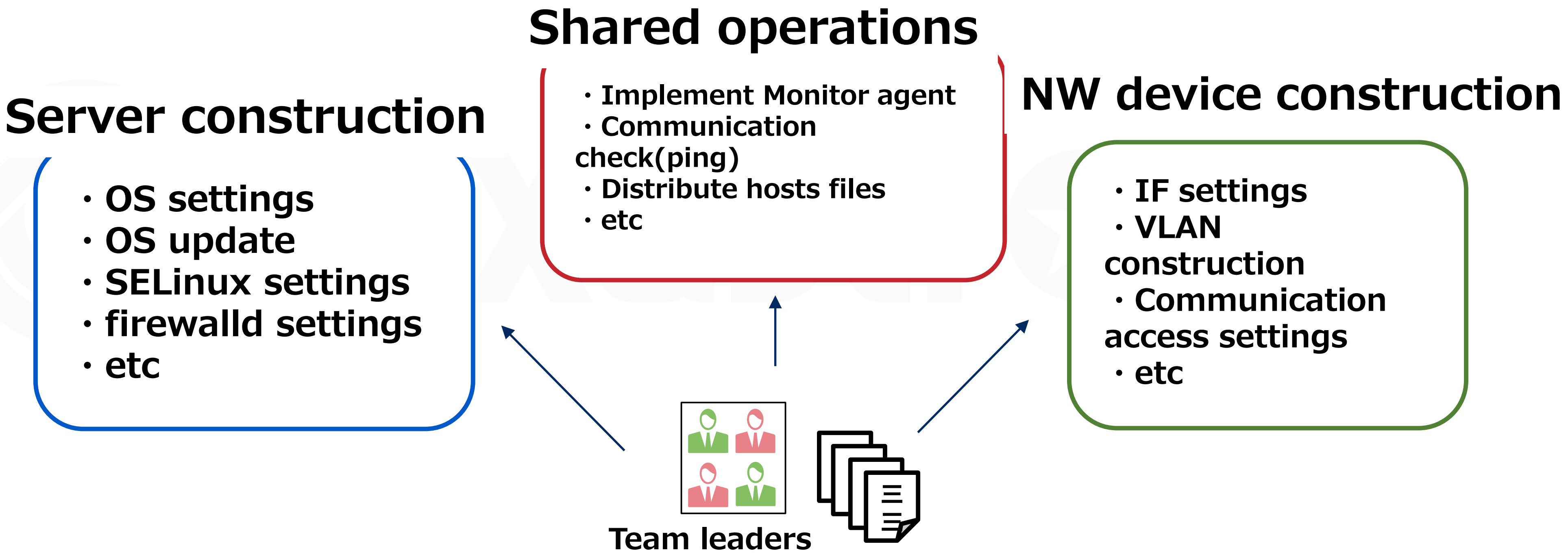
Step 2 : Actualize Automatic Execution

Tasks

- **Classify automated tasks**
- Make Operations more detailed
- Prepare Ansible files (Playbooks, etc.)
- Construct Job flow (Conductor)
- Execute Job flow (Conductor) Parameters are registered manually

Task explanation

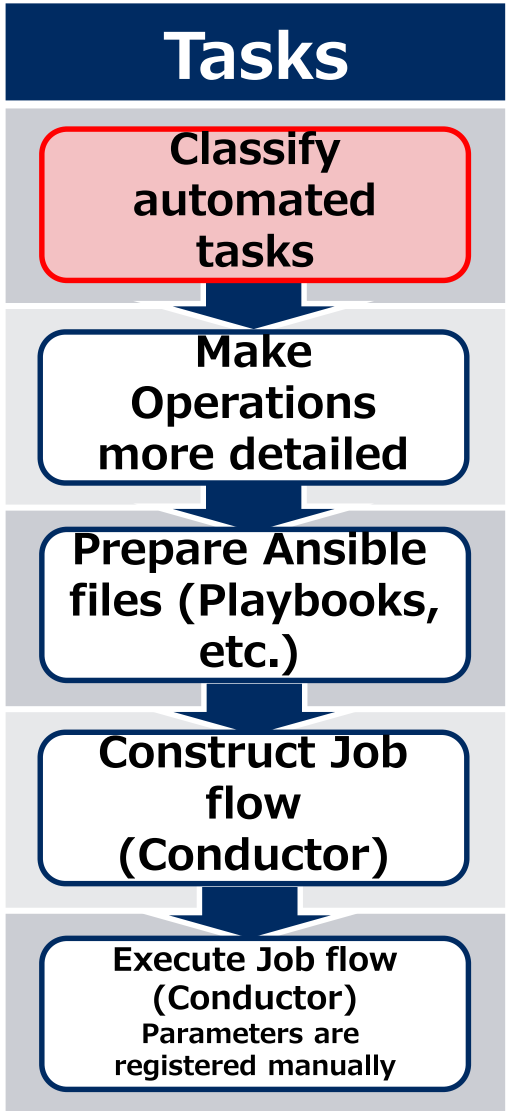
Organize the manually executed tasks and select which ones to automate.
If the organized tasks crosses more than one team, the team leaders will do the coordination.



POINT
① Categorize tasks with “just right” granularity.
② Estimate the effects of the operation and arrange them by priority.

Check next page

Step 2 : Actualize Automatic Execution

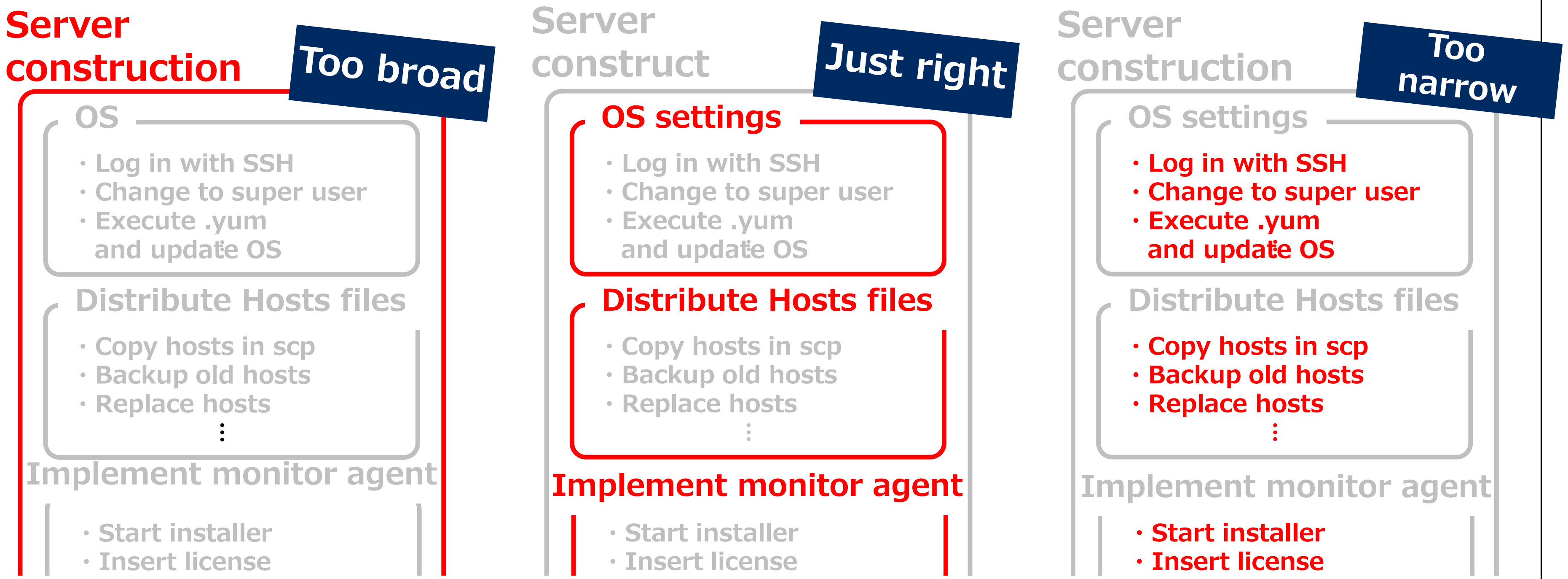


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor) Parameters are registered manually

POINT ① Categorize tasks with "just right" granularity.

Categorize the tasks that are getting automated with "just right" granularity. For example, for server construction, the example in the bottom right has too much information. On the other hand, the one on the left is too broad.

As can be seen in the middle figure, the "OS Settings" illustrates the perfect amount of granularity.



Step 2 : Actualize Automatic Execution

Tasks

- **Classify automated tasks**
- **Make Operations more detailed**
- **Prepare Ansible files (Playbooks, etc.)**
- **Construct Job flow (Conductor)**
- **Execute Job flow (Conductor)** Parameters are registered manually

POINT ② Estimate the effects of the operation and arrange them by priority

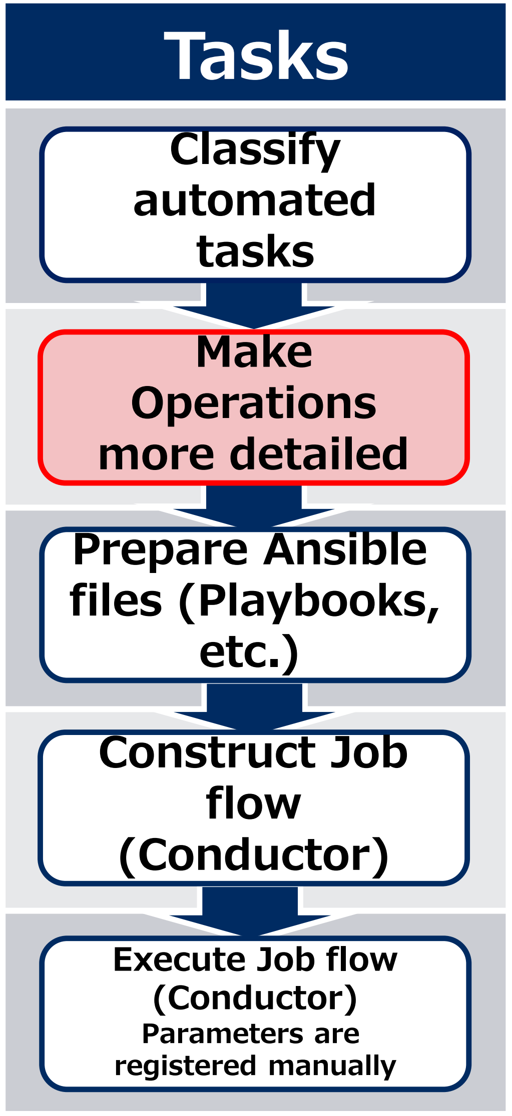
Estimate the effects of the operations and arrange them by priority. Once we know the effects, we can prioritize the tasks and decide whether to automate them or not.

Estimated effects includes the number of times the operation is used per year, the number of target devices and the number of man-hours per project. If the number isn't a quantitative number, it is possible to sort them by "Large", "Medium", or "Small". The following is an example of an organized list of operations with priority.

Operation	Times used	Number of devices	Man-hour per worker	Man-hour	Priority	Remarks
OS settings	50	50	10H	5H	**High**	Requires 2 persons
Distribute Hosts files	200	50	1H	0.5H	**Middle**	Updates 4 times a year
Implement monitor agent	30	30	5H	5H	**Low**	
Update Web contents	600	5	1H	1H	**High**	Updates 10 times a month
Summarize Access log	60	5	2H	2H	**Low**	Executed at the end of the month

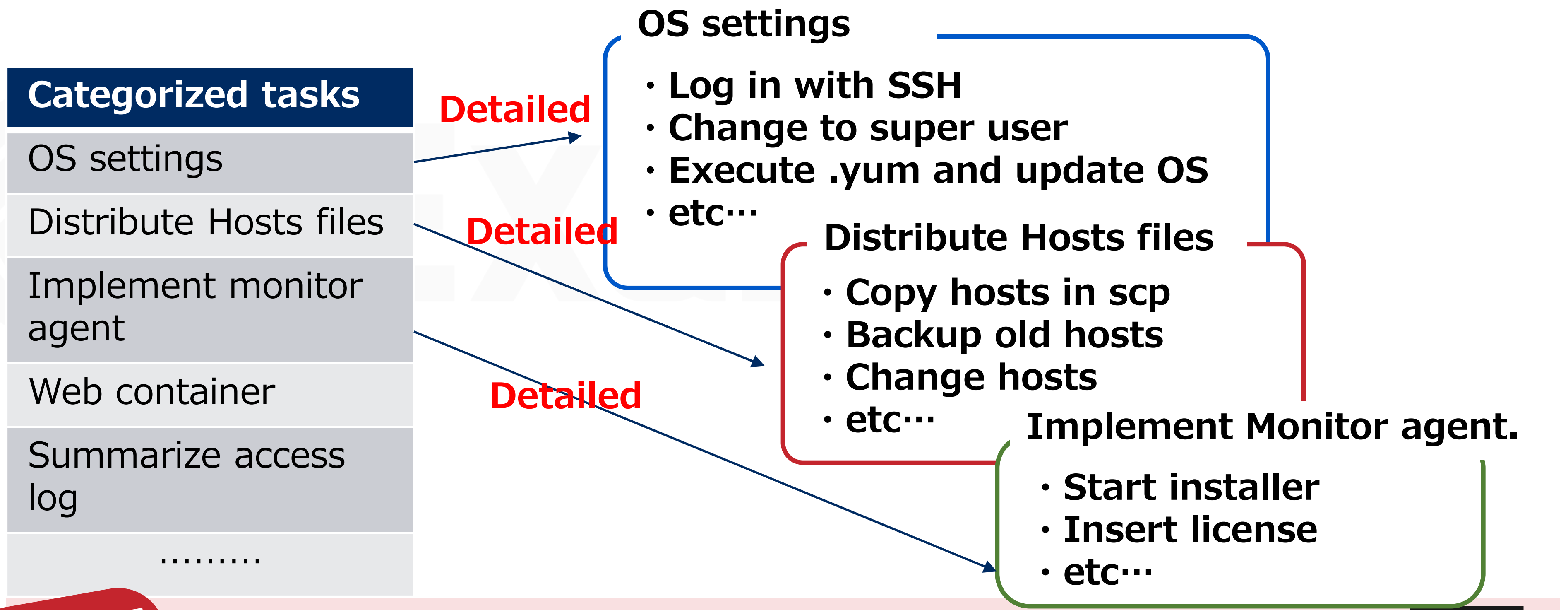
As a general rule, automation tends to be more effective for common tasks, since they are used more often per year. Additionally, by reviewing the granularity of the tasks, we can find out which tasks are common.

Step 2 : Actualize Automatic Execution


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor)
Parameters are registered manually

Task explanation

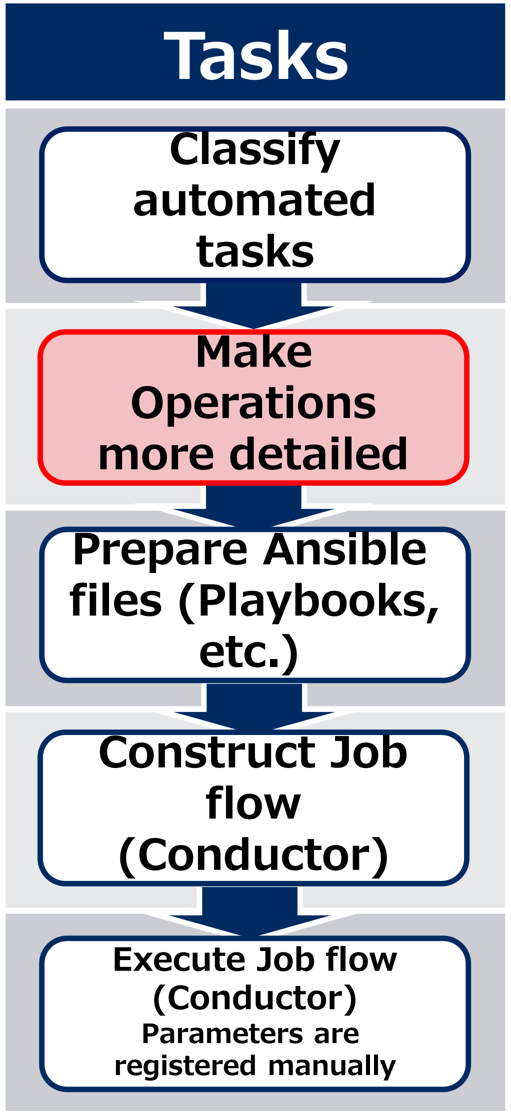
Make the categorized tasks more detailed and reduce them to more specific operations.
Detailing operations can be based on existing procedures and other documents.



POINT **①Backup, Run operation and Acquire backup.**

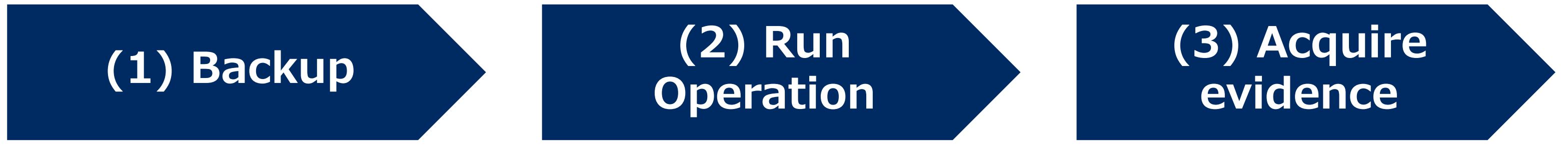
Check next page

Step 2 : Actualize Automatic Execution


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor)
Parameters are registered manually

POINT ① Backup, Run operation and Acquire backup.

We recommend to structure the detailed operations in sets of 3

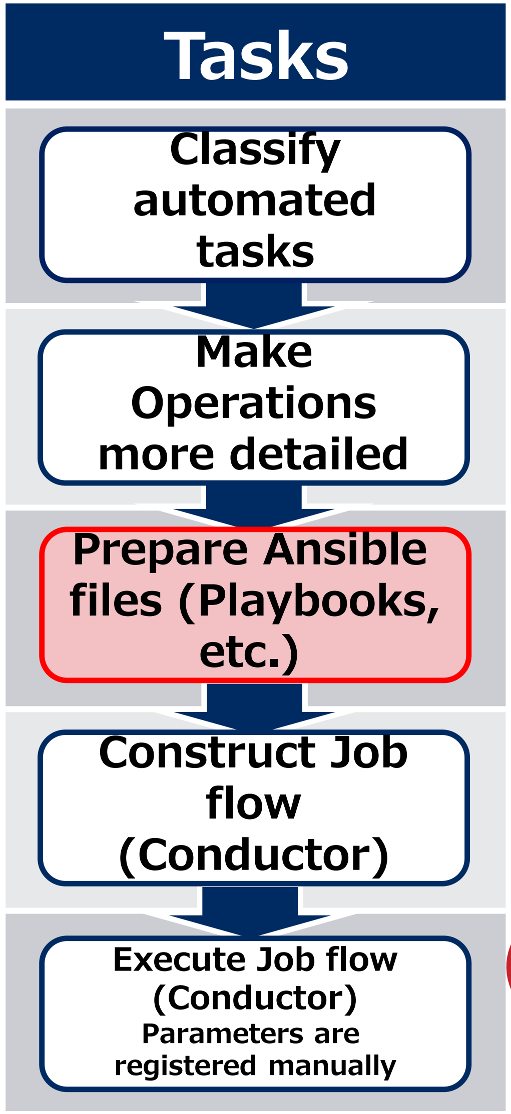


This configuration ensures that backups and evidence are available at all times, meaning that the operations can be safely re-used.

As an example, the following is the configuration of a procedure that distributes hosts files.

Process	Specific procedure
(1) Backup	Takes back up of current hosts files.
(2) Run operation	Copies new host files to the designated place.
(3) Acquire evidence	Saves successful name resolution results.

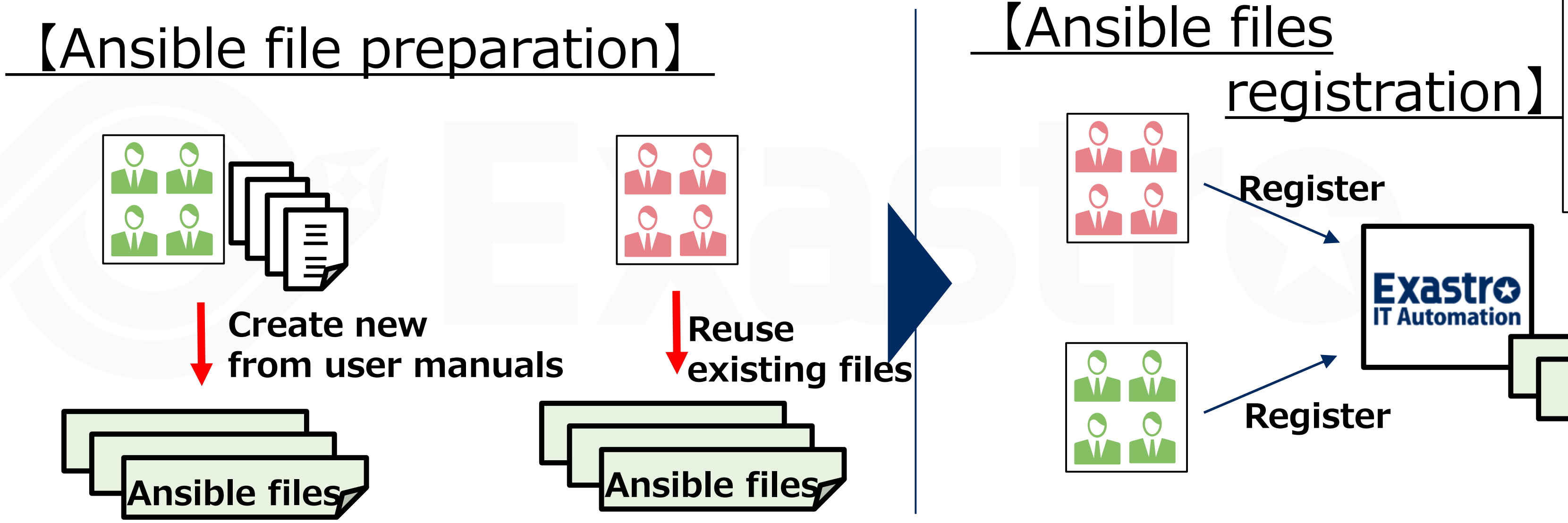
Step 2 : Actualize Automatic Execution


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor) Parameters are registered manually

Task explanation

Prepare Ansible files (Playbook, Etc.) to execute the procedure. You can create new one or use existing ones.

Ansible files is a set of files required for an operation to run.
・Playbook
・Role
・File
・Template

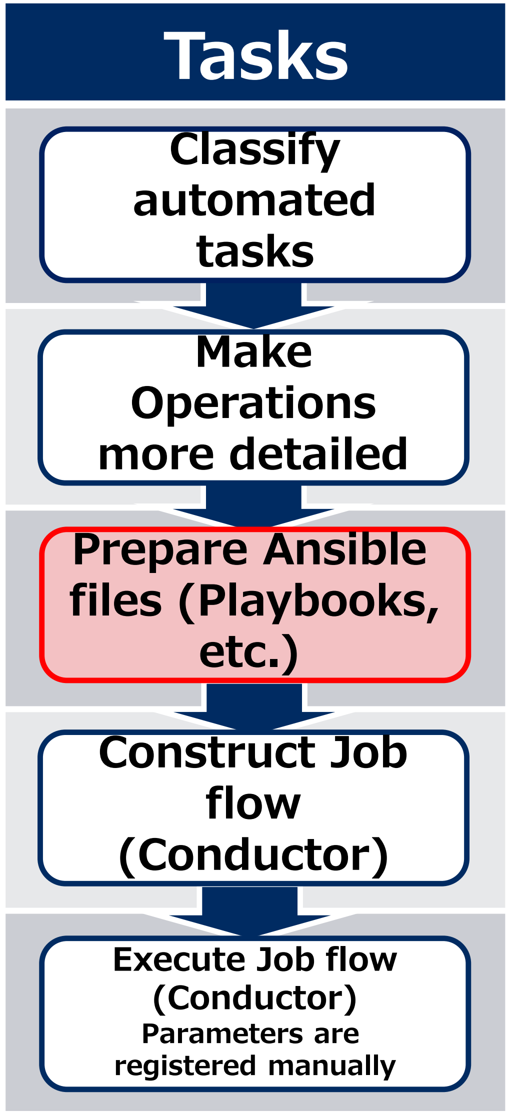


POINT

① Reuse any existing files available
② Variablize any values that changes for each operation run.
③ Keep similar processes concise by repeating.
④ Create a standard configuration for templates.

Check next page

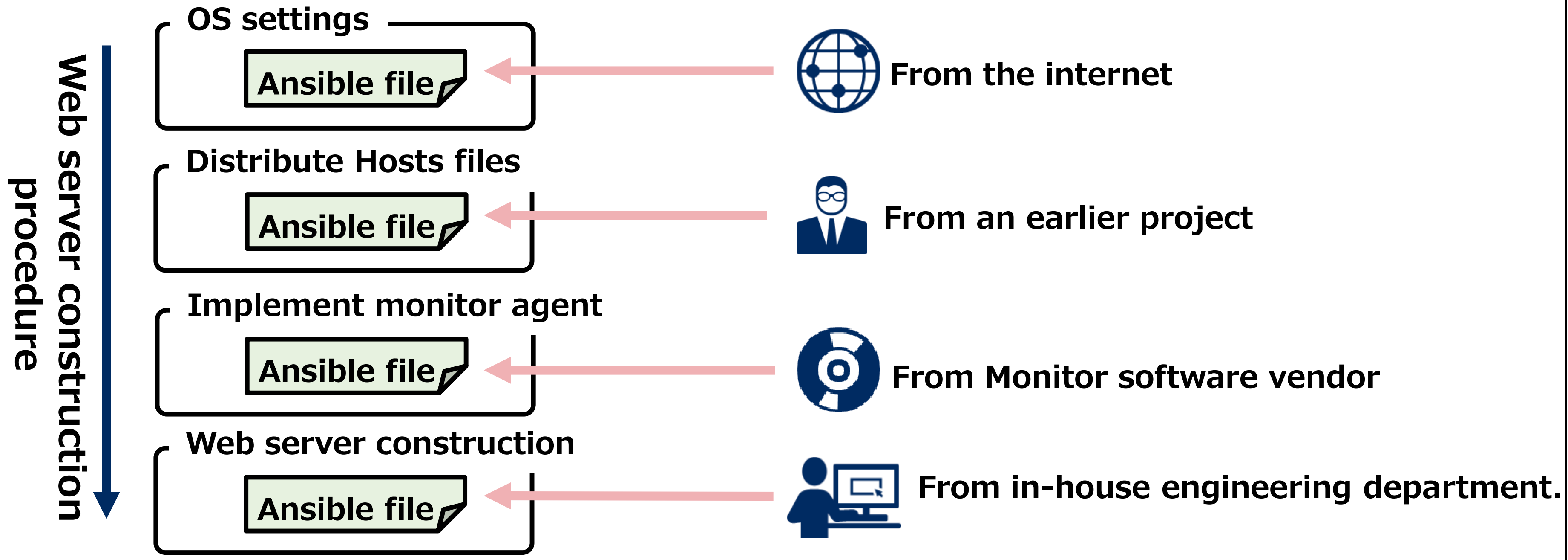
Step 2 : Actualize Automatic Execution


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor)
Parameters are registered manually

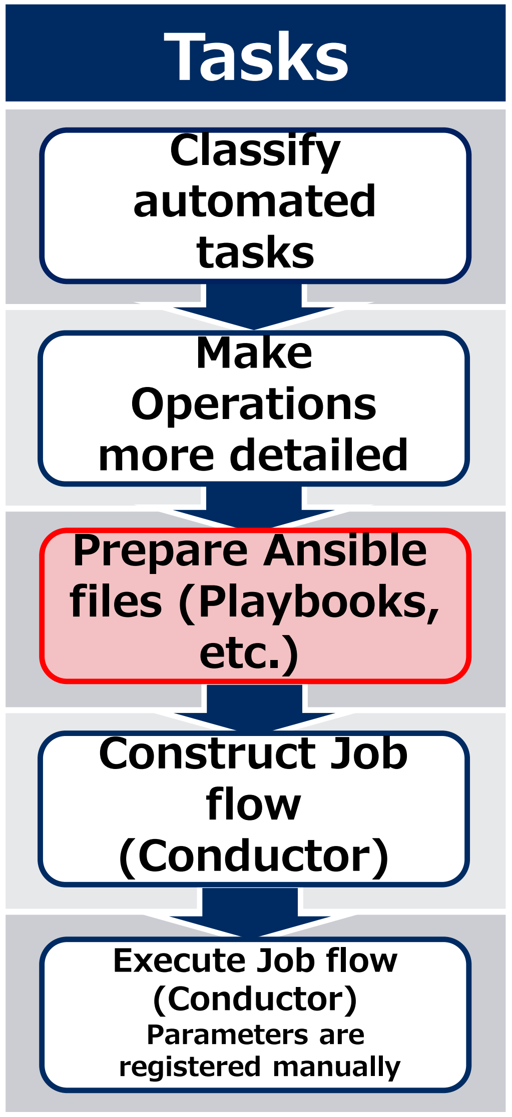
POINT ① Reuse any existing files available

You don’t need to create every part manually in an Ansible file. If you have any existing files, it is possible to use parts of them to create other files more efficiently.

The following example illustrates how to build a web server by using Ansible files from various sources.



Step 2 : Actualize Automatic Execution


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor) Parameters are registered manually

POINT **② Variablize any values that changes for each operation run.**

Some values, such as the host name for the machine, will change when the operations are executed. If you embed these values as fixed values in the Ansible files, you will need to modify the files every time you run an operation.

In order this, we use "variables" in Ansible files.

Playbook before variablization

```
- hostname:
    name: web01
```

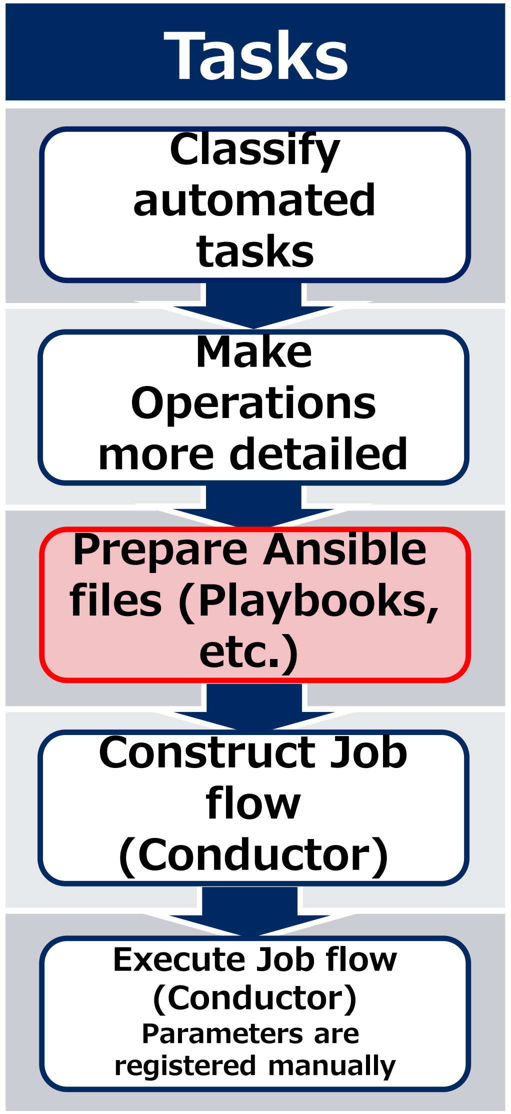
Playbook after variablization

```
- hostname:
    name: {{ VAR_hostname}}
```

The playbook on the left has a fixed host name,"web01". If we don't change it, we will need to modify the playbook in order to set up "web02" on another machine.

On the other hand, the playbook on the right has the host name converted into a variable, {{ VAR_hostname }}. By setting specific values for the variables separately, the variablized parts can be replaced with any expected values when the operation is executed.

Step 2 : Actualize Automatic Execution


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor)
Parameters are registered manually

POINT ③ Keep similar processes concise by repeating.

If the tasks are organized to be executed automatically, you might see that some similar tasks are used multiple times. In those cases, we can keep the process concise by using repetition. In the case of Ansible's Playbooks, we can use the "Loop" instruction.

The following is an example of a playbook that creates three directories: /dir1, /dir2 and /dir3. The playbook on the left runs 3 different processes. On the other hand, the one on the right uses "loop" to repeat the process, which makes it more concise and easier to maintain.

Not repeated playbook

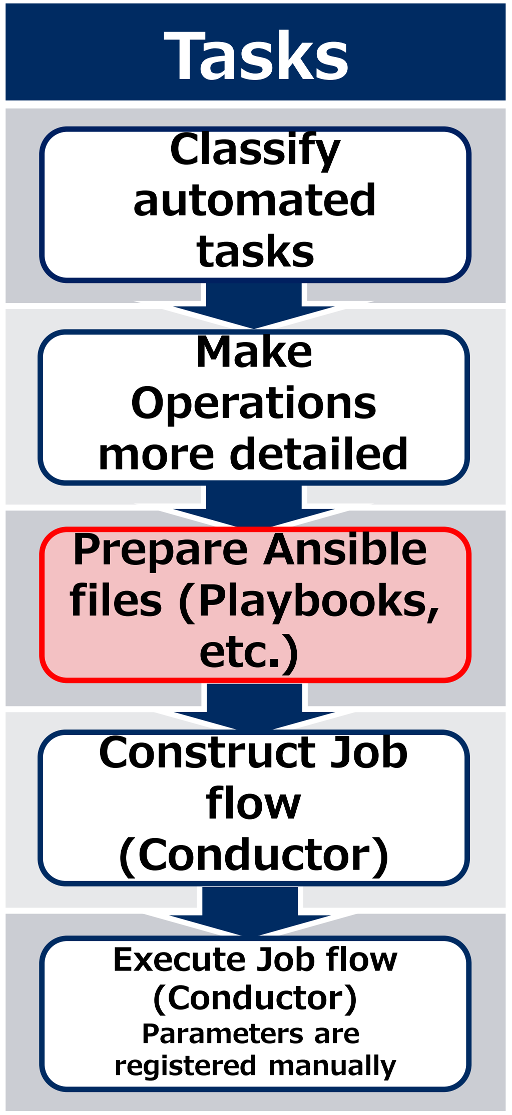
```
- file:
    path: /dir1
    state: directory

- file:
    path: /dir2
    state: directory

- file:
    path: /dir3
    state: directory
```

Repeat

Repeated playbook

```
- file:
    path: ″{{ item }}″
    state: directory
  loop: {{ VAR_dirs }}
```

Step 2 : Actualize Automatic Execution



Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor) Parameters are registered manually

POINT ④ **Create a standard configuration for templates.**

In situations where setting files are distributed to multiple servers, the contents of the files are in many cases almost the same, which only some of the values being different. In these cases, we can be more efficient by creating setting files using formats.

In Ansible, Files with .j2 extensions are "Format" files. Similarly to playbooks, formats can also use variables. The following is an example of an Apache settings file being created. The blue text are variables and the red text are values after it has been created.

httpd.conf.j2 (Format)

```
<VirtualHost *:80>
  ServerName {{ VAR_hostname }}
  DocumentRoot {{ VAR_docroot }}
</VirtualHost>
```

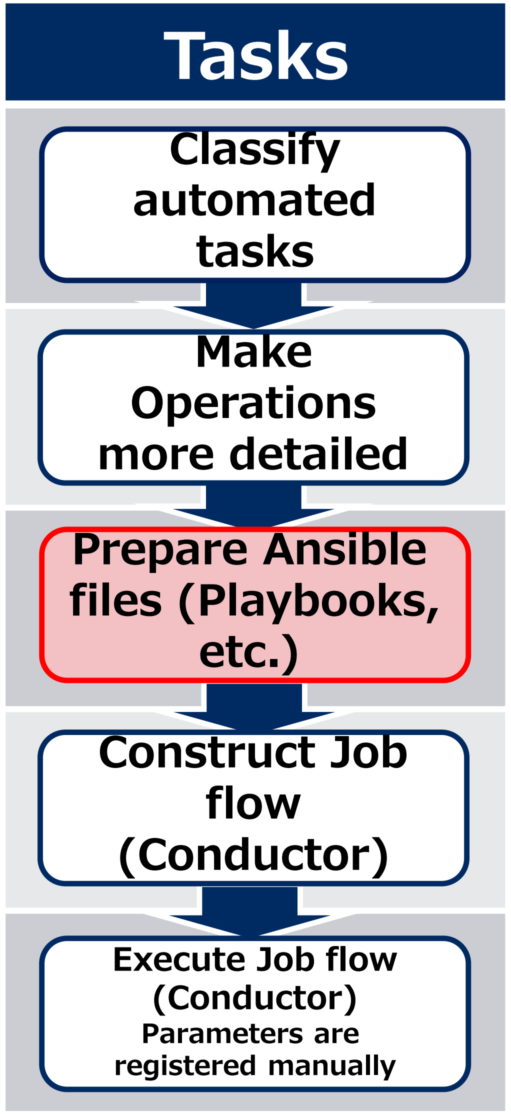
Create

```
<VirtualHost *:80>
  ServerName www.test.com
  DocumentRoot /contents
</VirtualHost>
```

Create

```
<VirtualHost *:80>
  ServerName www.dev.com
  DocumentRoot /public
</VirtualHost>
```

Step 2 : Actualize Automatic Execution


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor) Parameters are registered manually

Appendix : Managing Playbooks

This section describes how to manage Ansible materials (Playbooks,etc.), using problems and solutions that actually happened as examples.

PROBLEM

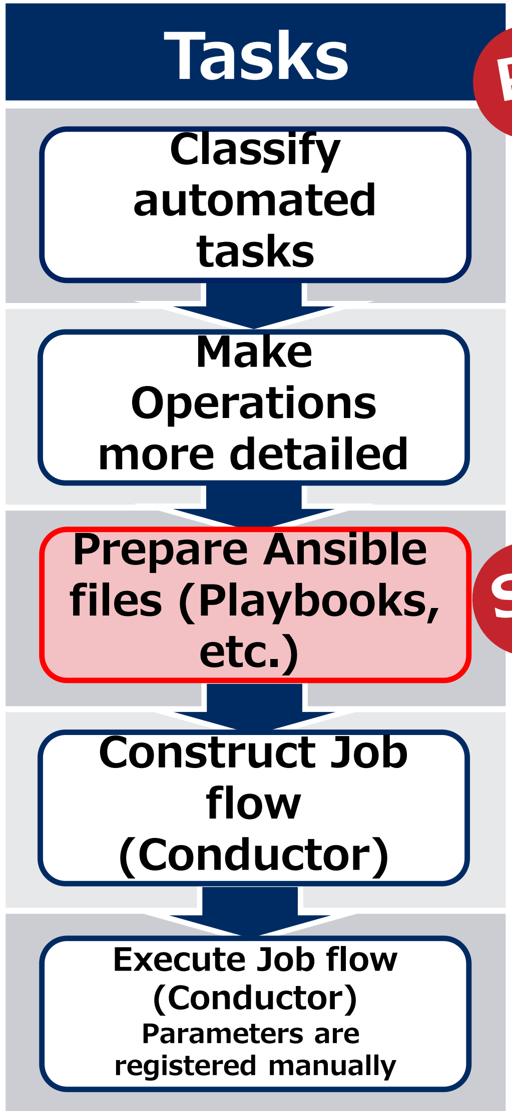
① The same playbook exists across multiple directories.
② Playbooks with different contents have the same name.
③ There are differences in playbook contents between the version management tool and ITA.

SOLUTION

① Create a directory for common processes.
② Decide on a naming convention in advance and don’t allow files with same name
③ Manage using a version management tool and CICD tool.

Check next page

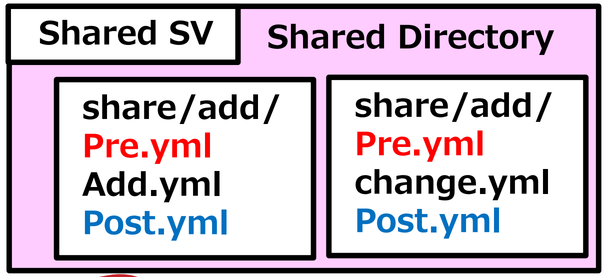
Step 2 : Actualize Automatic Execution


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor) Parameters are registered manually

PROBLEM

① The same playbook exists across multiple directories.

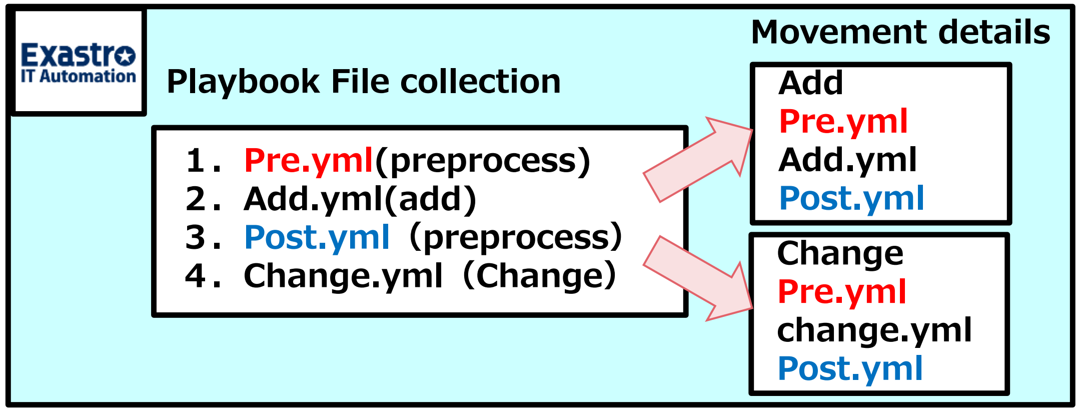
When we managed a shared directory, we created a new directory for each process, causing files to exist over multiple directories.



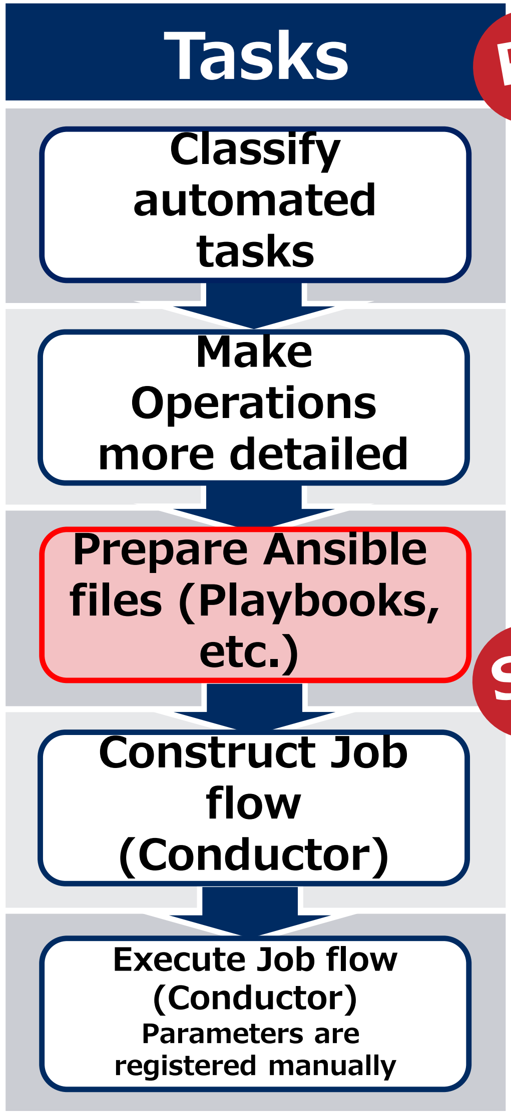
SOLUTION

① Use one Playbook for multiple processes.

ITA allows users to manage Playbooks in one central place, making it possible to use the same Playbook in different Movements.

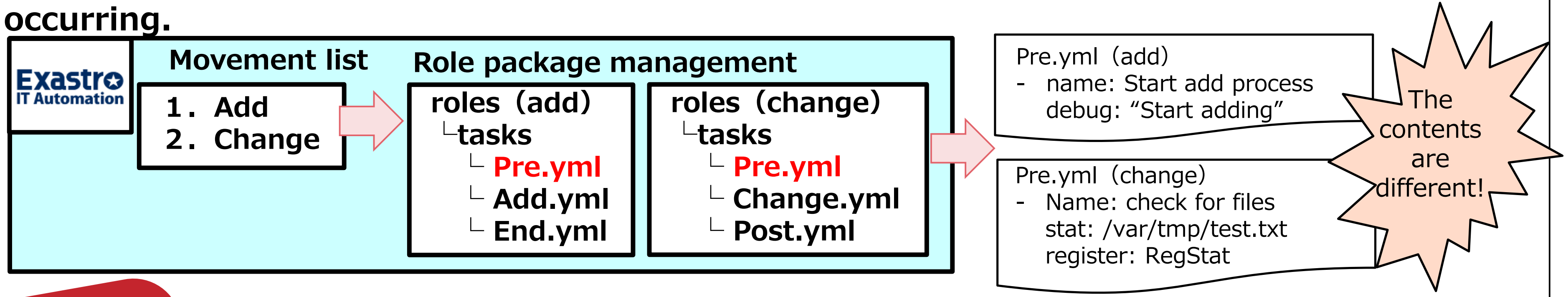


Step 2 : Actualize Automatic Execution


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor) Parameters are registered manually

PROBLEM ② **Playbooks with different contents have the same name.**

Two files with the same name but different contents was accidentally created in Ansible Legacy Role.
Therefore, altering the "add" Pre.yml also changes the "change" Pre.yml, leading to a bug occurring.

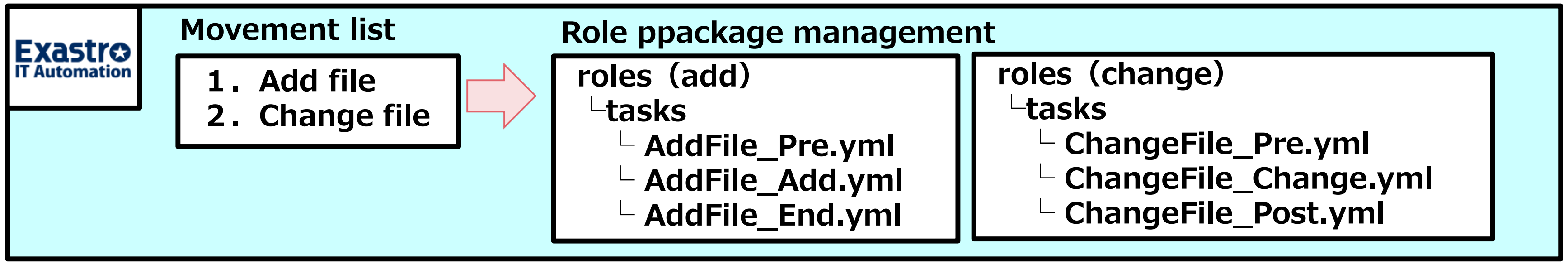


```
Pre.yml (add)
-  name: Start add process
   debug: "Start adding"
```

```
Pre.yml (change)
-  Name: check for files
   stat: /var/tmp/test.txt
   register: RegStat
```

SOLUTION ② **Decide on a naming convention in advance and don't allow files with same name**

Ansible Role allows for files with same name but different packages.
However, as this often leads to bugs, we recommend deciding on a naming convention and forbidding files with same name.
Example: **Playbooks are named in this format "Process_XXX.yml"**



Step 2 : Actualize Automatic Execution

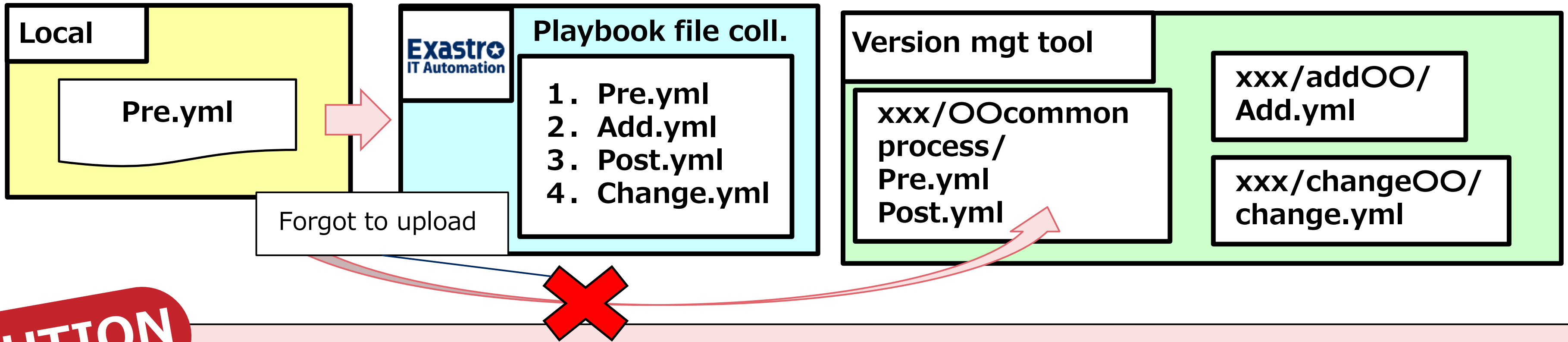
Tasks

- Classify automated tasks
- Make Operations more detailed
- Prepare Ansible files (Playbooks, etc.)
- Construct Job flow (Conductor)
- Execute Job flow (Conductor) Parameters are registered manually

PROBLEM

③ There are differences in playbook contents between the Version management tool and ITA.

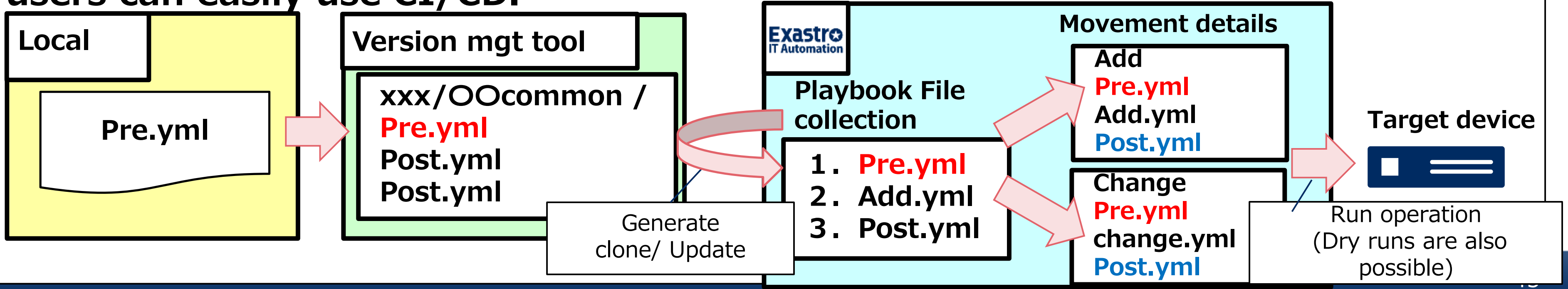
When adding and repairing Playbooks, we upload them both to ITA and a version management tool (Git, and such), but I forgot to upload it to ITA, meaning that the fix/new one wont get displayed.



SOLUTION

③ Manage using ITA's CI/CD for IaC function.

For cases like these, we recommend that you use the ITA CI/CD for IaC function. This function automatically updates the files uploaded to ITA when the files in the version management tools are updated. If linked together with a Movement, users can easily use CI/CD.



Step 2 : Actualize Automatic Execution

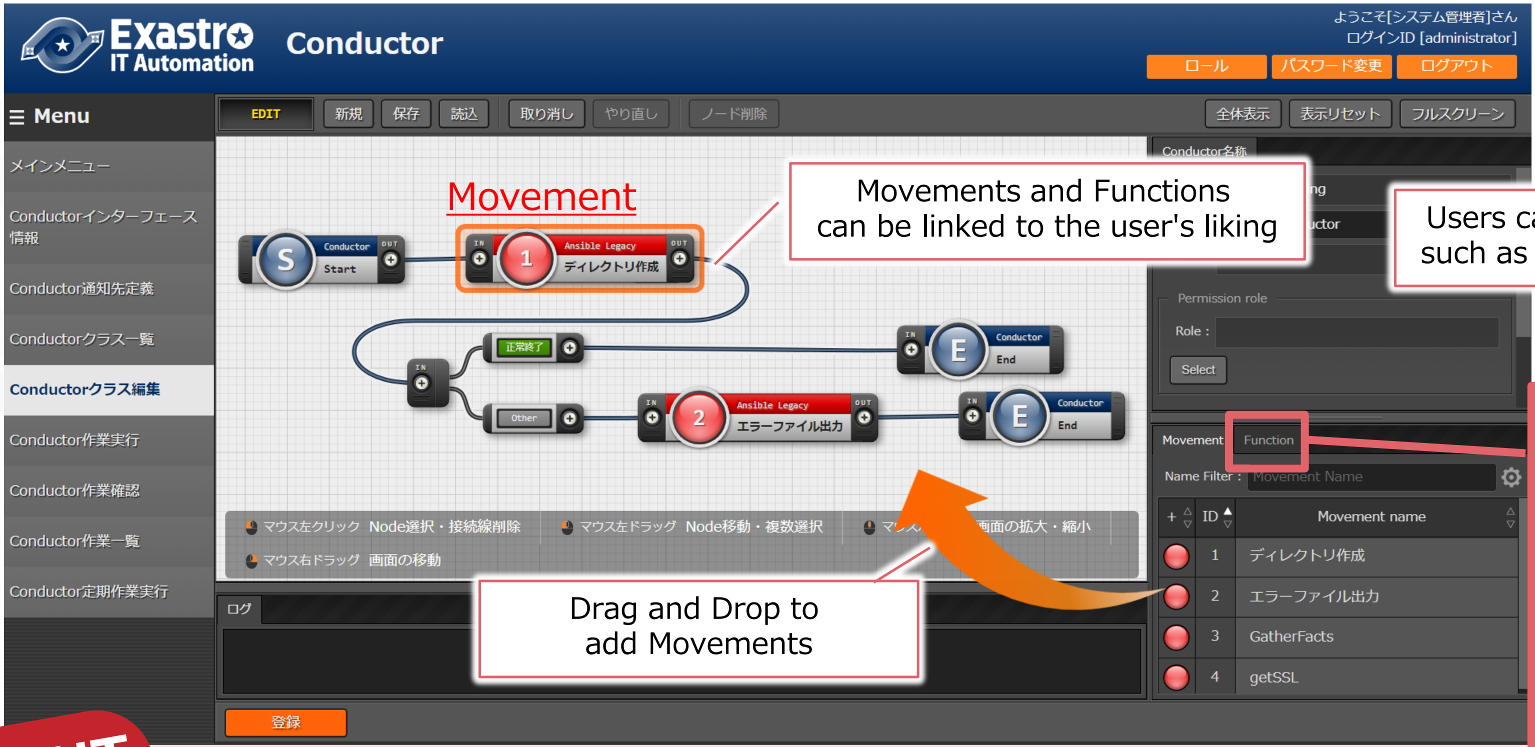
Tasks

- Classify automated tasks
- Make Operations more detailed
- Prepare Ansible files (Playbooks, etc.)
- **Construct Job flow (Conductor)**
- Execute Job flow (Conductor) Parameters are registered manually

Task explanation

Create a Jobflow in IT Automation.

【Jobflow Creation screen】

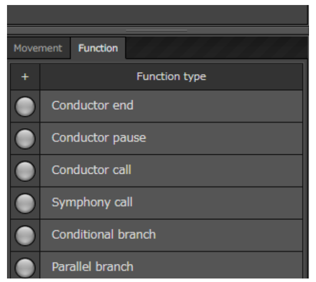


Movement

Movements and Functions can be linked to the user's liking

Users can also use functions such as Conditional branches.

Function

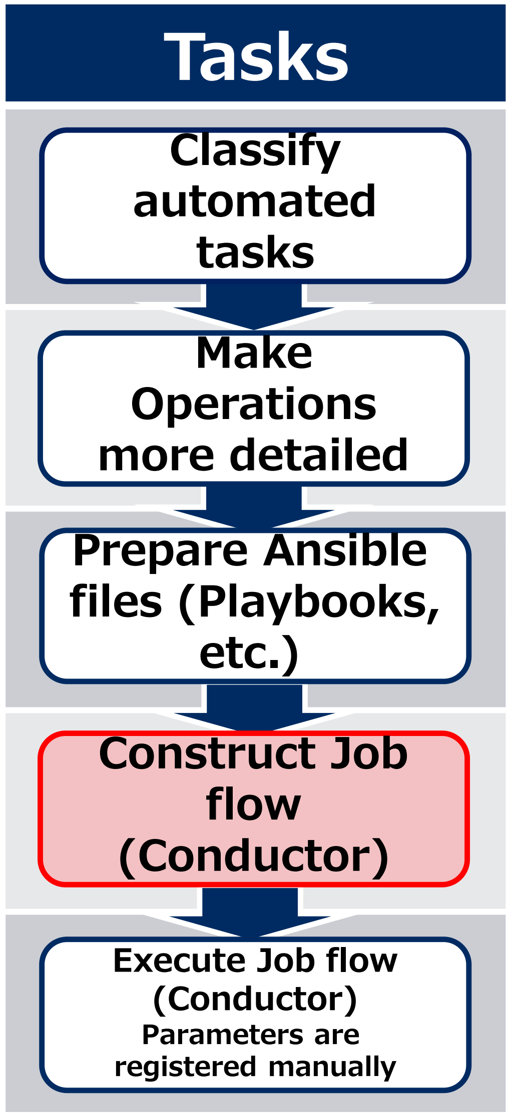


Drag and Drop to add Movements

POINT ① **Understand the process of creating Jobs and Jobflows.**

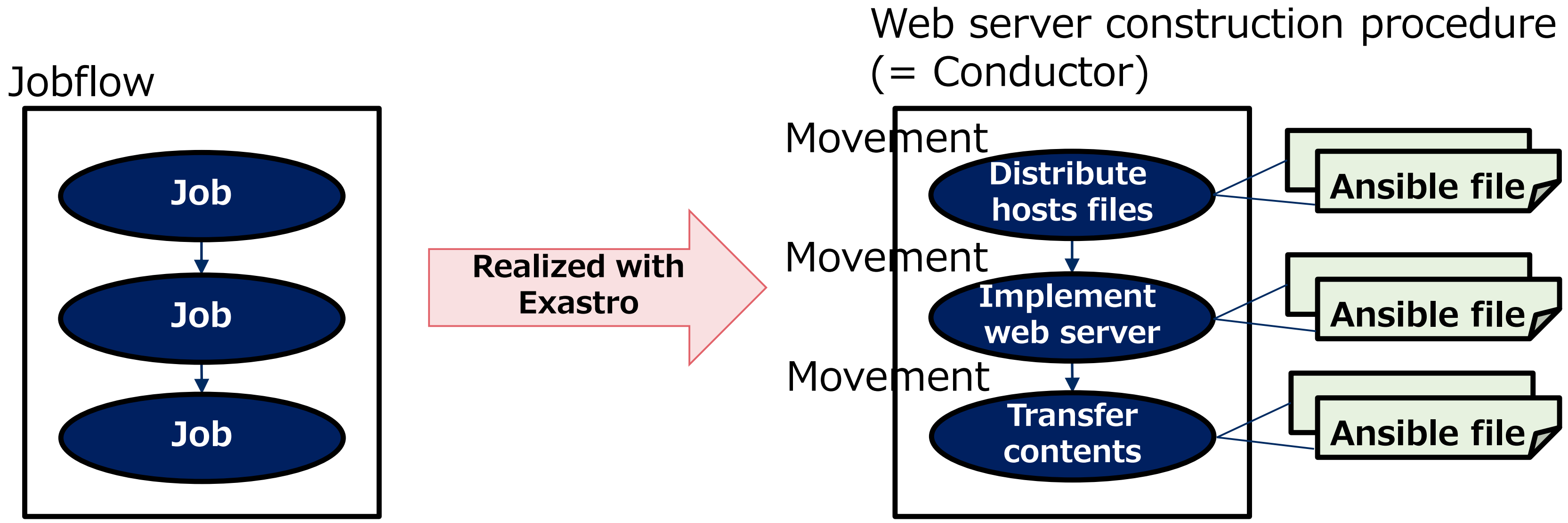
Check next page

Step 2 : Actualize Automatic Execution


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor)
Parameters are registered manually

POINT ① **Understand the process of creating Jobs and Jobflows.**

The operations that we categorized in the first task of step 2, Classifying Automated Tasks, is called a "job". A "Jobflow" is a string of several jobs that are executed in a specific order.



In Exastro IT Automation, jobflows are made possible with the "Conductor" function, and "Jobs" by the "Movement" function.
By linking an Ansible file (Playbook, etc.) to a movement, it becomes possible to run operations with real effects.

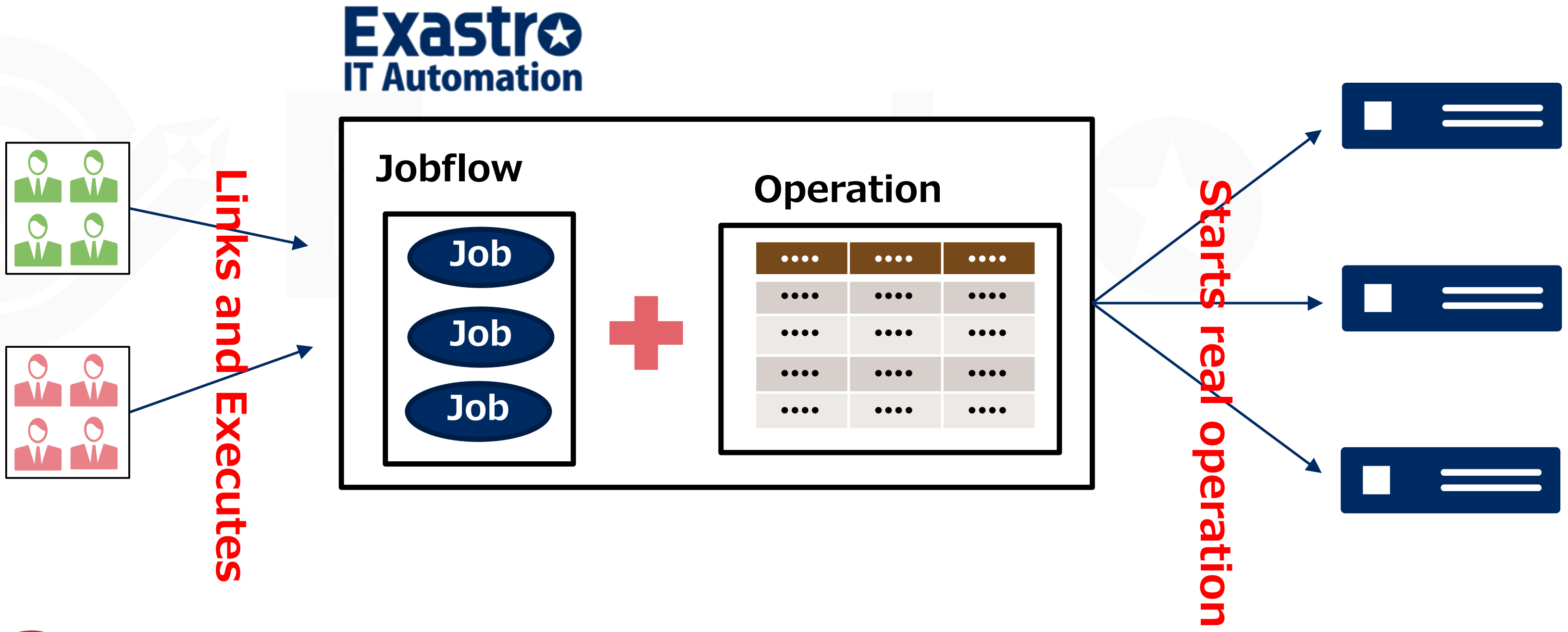
Step 2 : Actualize Automatic Execution

Tasks

- Classify automated tasks
- Make Operations more detailed
- Prepare Ansible files (Playbooks, etc.)
- Construct Job flow (Conductor)
- Execute Job flow (Conductor) Parameters are registered manually

Task explanation

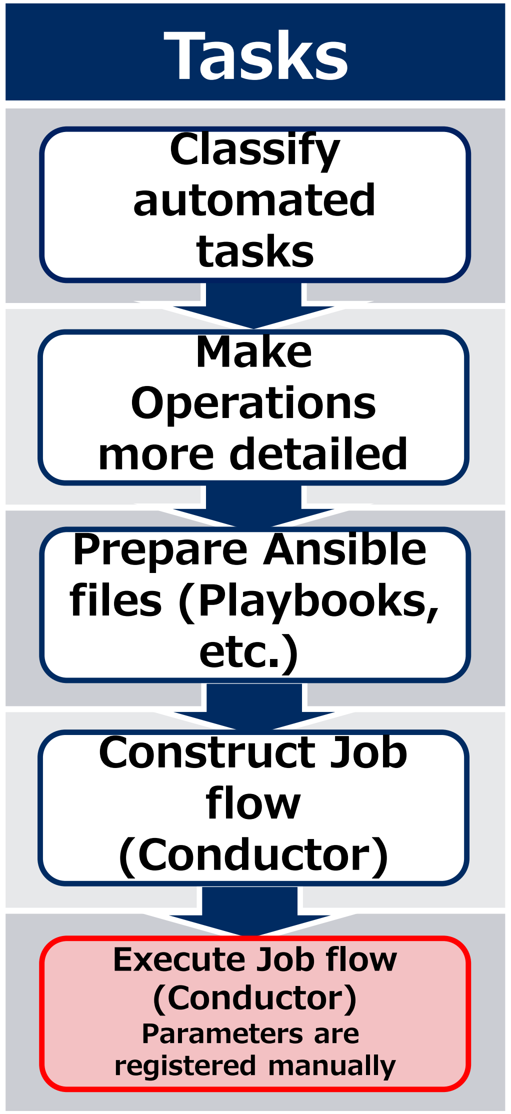
Link Jobflow and Operation and Automatically execute the Operation.



POINT ①Understand the relationship between Operations and Jobflows

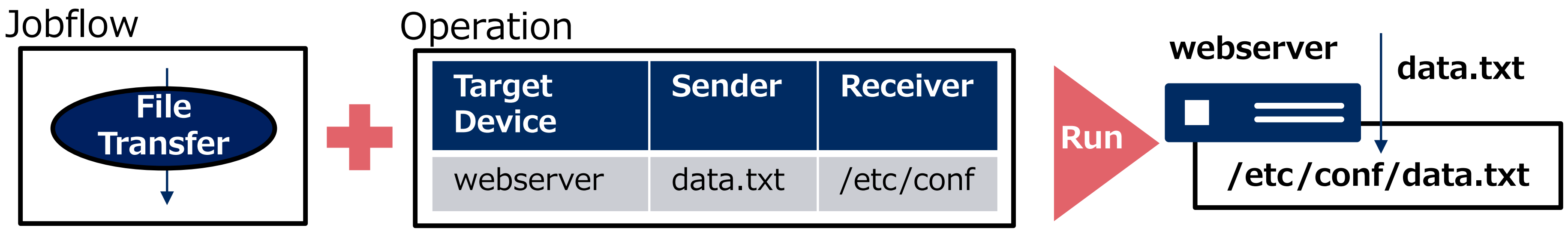
Check next page

Step 2 : Actualize Automatic Execution


Tasks
Classify automated tasks
Make Operations more detailed
Prepare Ansible files (Playbooks, etc.)
Construct Job flow (Conductor)
Execute Job flow (Conductor)
Parameters are registered manually

POINT ①Understand the relationship between Operations and Jobflows

An Operation links a target device and specific setting values to a Jobflow. The following illustrates a simple Jobflow that transfers files to a server.



Target Device	Sender	Receiver
webserver	data.txt	/etc/conf

With the help of the Operation, “Target Device” ,”Sender” and “Receiver” gets linked to the Jobflow. The combination above deploys Data.txt to the web server.

By changing the inside of the Operation, we can choose to send different files to different target devices.

Automation Preparation

Step 1 : Central management of the Configuration info.

Step 2 : Actualize Automatic Execution.

Step 3 : Linking Central management and automatic execution.

Step 3： Linking Central management and automatic execution

The following slides explains the 2 tasks in step 3.

Design

☑ Delays and … mmunicating between teams.

☑ Doubl… ma… tary wording leads to errors in the design

… de… plications with managing design documents (forms)

☑ As a result … and af…

Preparation

☑ Work order … t

gets discarded.

☑ Every opera… er i s cr…

☑ Configurati… cedure,

i… e… lded (ba

Execution

☑ Since … he c… he produ…

⇒People of … they can continue.

☑ Since most … one manually, human error is inevitable.

Solution

Solution

Link

Step 1

Centrally Manage conf…

Step …

Auto…

Job

Job

…cute

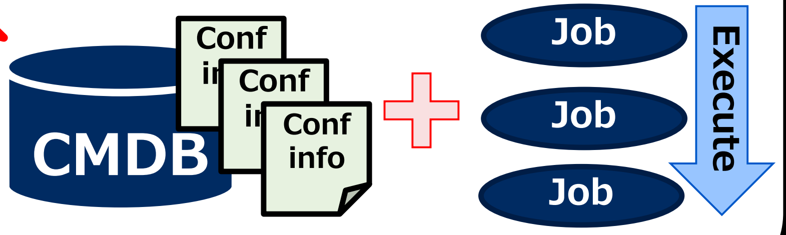
Tasks

Link Variable and Specific Value

Run Jobflow (Conductor)

Step 3

Linking Central management and automatic execution



Step 3： Linking Central management and automatic execution

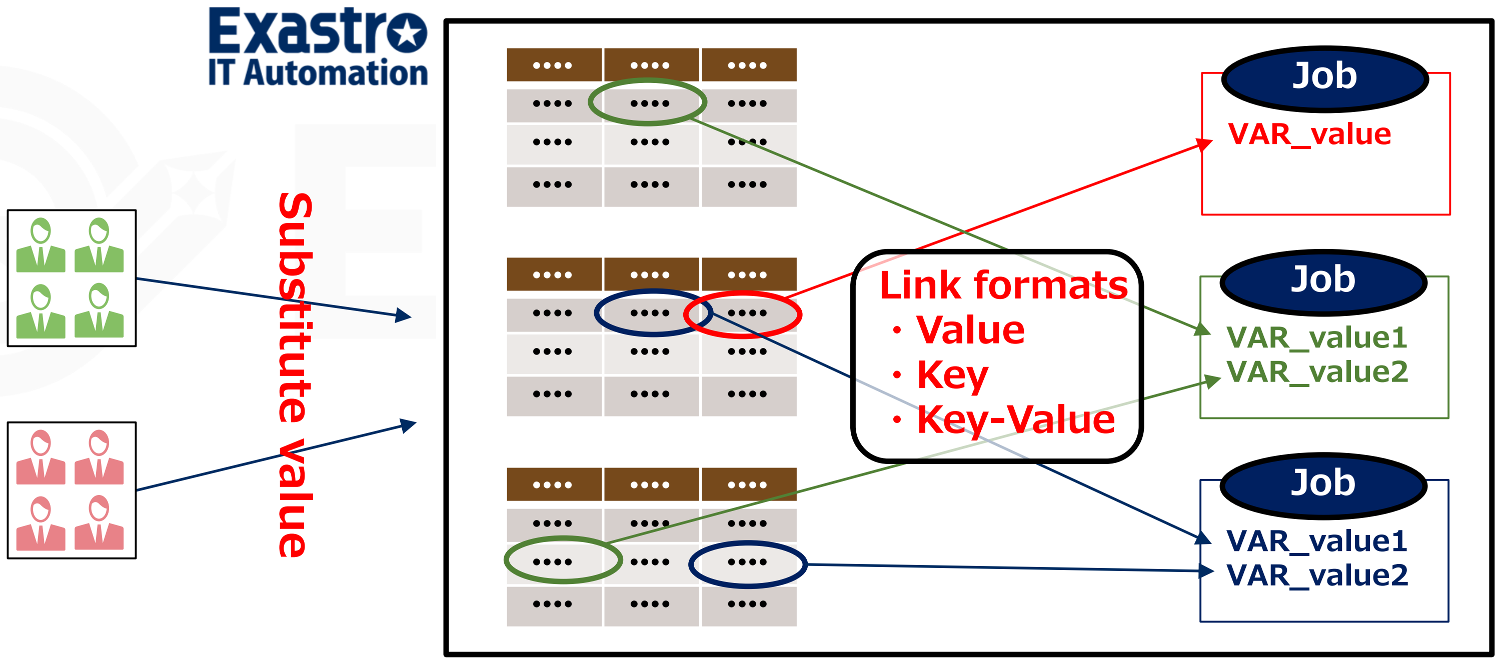
Tasks

Link Variable and Specific Value

Run Jobflow (Conductor)

Task explanation

Use the “Substitute automatic value registration list” function in IT Automation to link the parameter sheet values and the job variables.

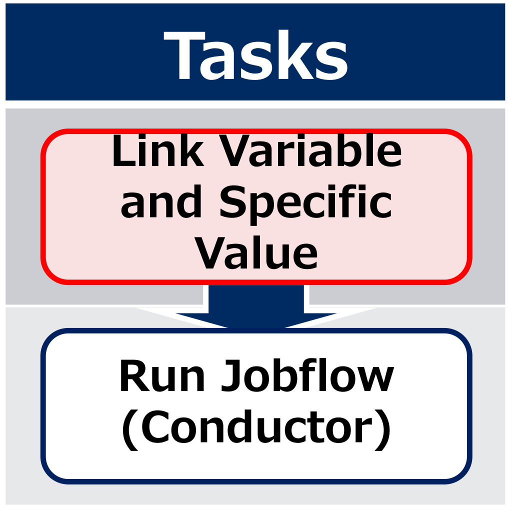


POINT
① **How to use Value-types**
② **How to use Key-types**
③ **How to use Key-Value types**

Check next page

Step 3： Linking Central management and automatic execution

Tasks

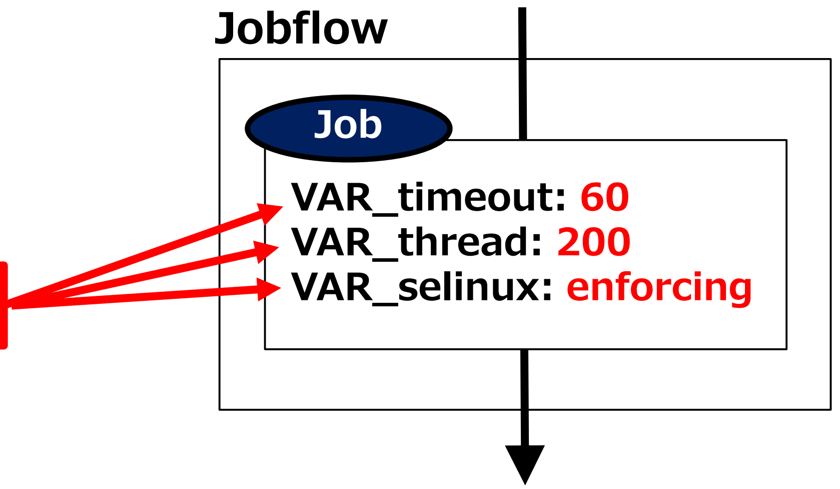


POINT ① **How to use Value-types**

Value type is a basic type and links the values inside the chart to the variables. It can be used for many things, such as for system settings and command line arguments.

The following illustrates how variables are linked to each of the server type settings.

Host name	Time out	Threads	SELinux
web1	60	200	enforcing
web2	60	200	enforcing
db-server	30	50	permissive



In the example above, each value in "web2" is linked with the job variables.

Step 3 : Linking Central management and automatic execution

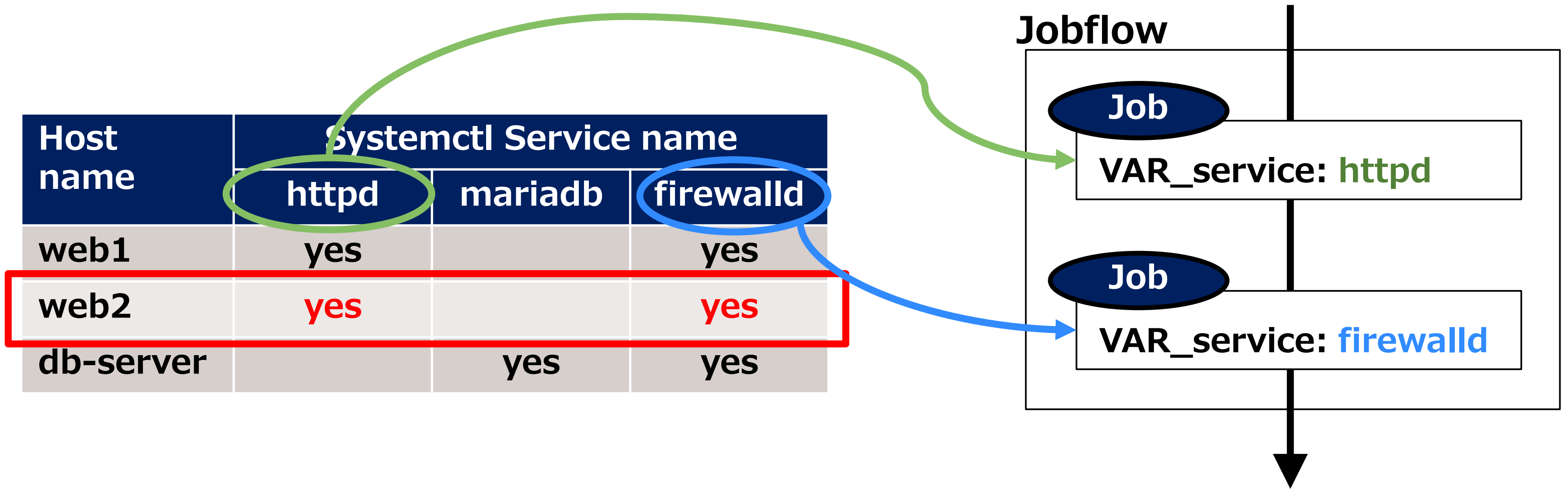
Tasks

- **Link Variable and Specific Value**
- **Run Jobflow (Conductor)**

POINT

② How to use Key-types

Key type is used to tie table column names to variables. It is mainly used as a flag. The following shows an example on how variables are linked to running services on a server.

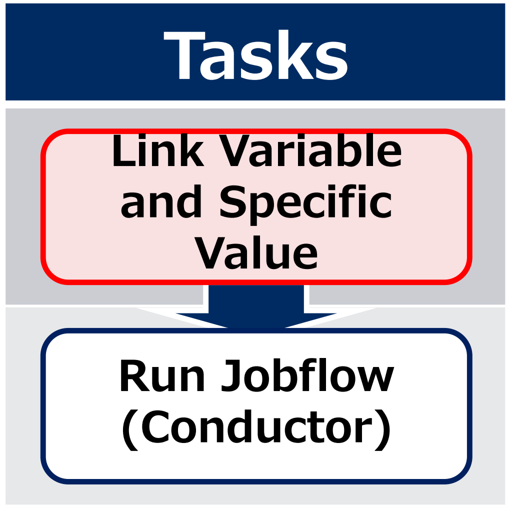


Host name	Systemctl Service name		
	httpd	mariadb	firewalld
web1	yes		yes
web2	yes		yes
db-server		yes	yes

In the example above, "Web2" has the columns, "httpd" and "firefalld" set to "yes", so the column names will be linked to the values of the variables and then execute the job.

Step 3： Linking Central management and automatic execution

Tasks

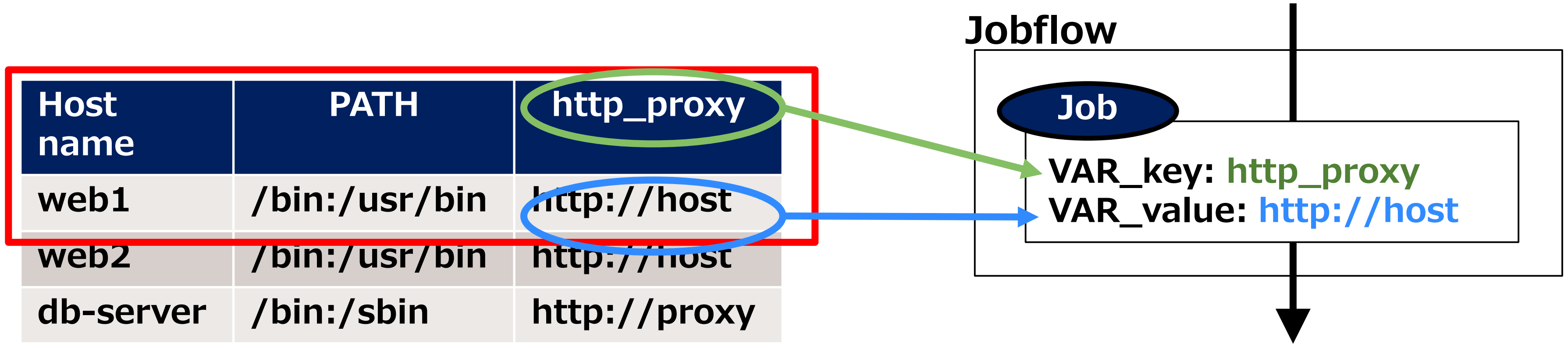


POINT

③ How to use Key-Value types

Key-Value types can be used to tie both they key and value to a variable. The following example shows how to set environment variables on the server using the Environment variable definition table.

Host name	PATH	http_proxy
web1	/bin:/usr/bin	http://host
web2	/bin:/usr/bin	http://host
db-server	/bin:/sbin	http://proxy



In the example above, the column name is the environment name.

Both the environment variable name, “http_proxy”, and it’s value ,”http://host” are linked to the variable.

Step 3： Linking Central management and automatic execution

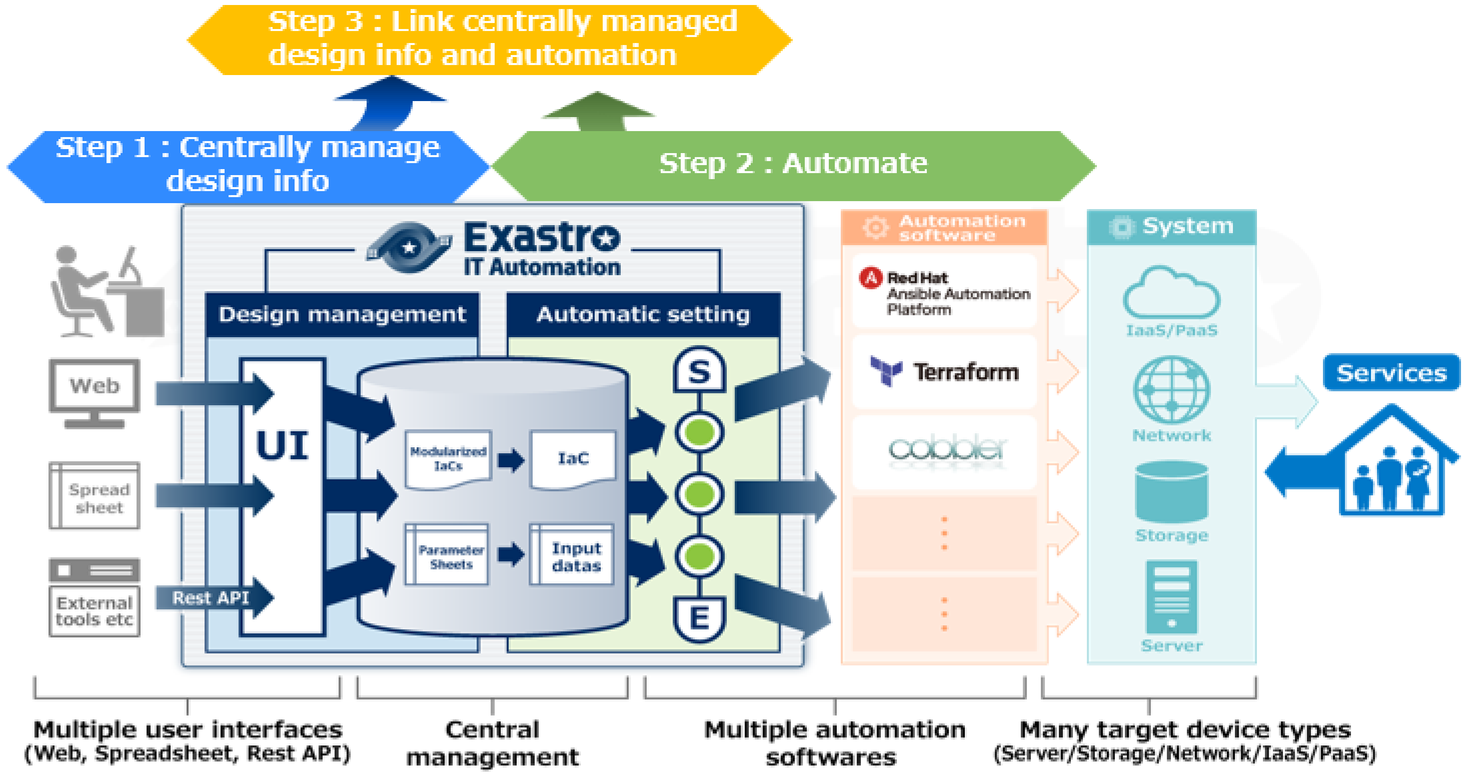
Tasks

Link Variable and Specific Value

Run Jobflow (Conductor)

Task explanation

Link Jobflow and Operation and automatically execute the operation.
Users can create systems by using these two actions:
Edit parameters→ Execute.


Step 3 : Link centrally managed design info and automation
Step 1 : Centrally manage design info
Step 2 : Automate
Exastro IT Automation
Design management
Automatic setting
Web
Spread sheet
External tools etc
Rest API
UI
Modularized IaCs
IaC
Parameter Sheets
Input datas
S
E
Automation software
Red Hat Ansible Automation Platform
Terraform
cobbler
System
IaaS/PaaS
Network
Storage
Server
Services
Multiple user interfaces (Web, Spreadsheet, Rest API)
Central management
Multiple automation softwares
Many target device types (Server/Storage/Network/IaaS/PaaS)

- Implementing automated SI
  - Effects and Estimations
  - Post-Automation Process changes and results.

- Implementing automated SI

  Effects and Estimations

  Post-Automation Process changes and results.

Exastro

Estimate the effects of the operation (repost)

Estimate the effects of the operations and arrange them by priority. Once we know the effects, we can prioritize the tasks and decide whether to automate them or not. Estimated effects includes the number of times the operation is used per year, the number of target devices and the number of man-hours per project.

Operation	Times used	Number of devices	Man-hour per worker	Man-hour	Priority	Remarks
OS settings	50	50	10H	5H	**High**	Requires 2 persons
Distribute Hosts files	200	50	1H	0.5H	**Middle**	Updates 4 times a year
Implement monitor agent	30	30	5H	5H	**Low**	
Update Web contents	600	5	1H	1H	**High**	Updates 10 times a month
Summarize Access log	60	5	2H	2H	**Low**	Executed at the end of the month

If the number isn't a quantitative number, it is possible to sort them by "Large", "Medium", or "Small". The following is an example of an organized list of operations with priority.

Case: Constructing Network Device(1/2)

Overview

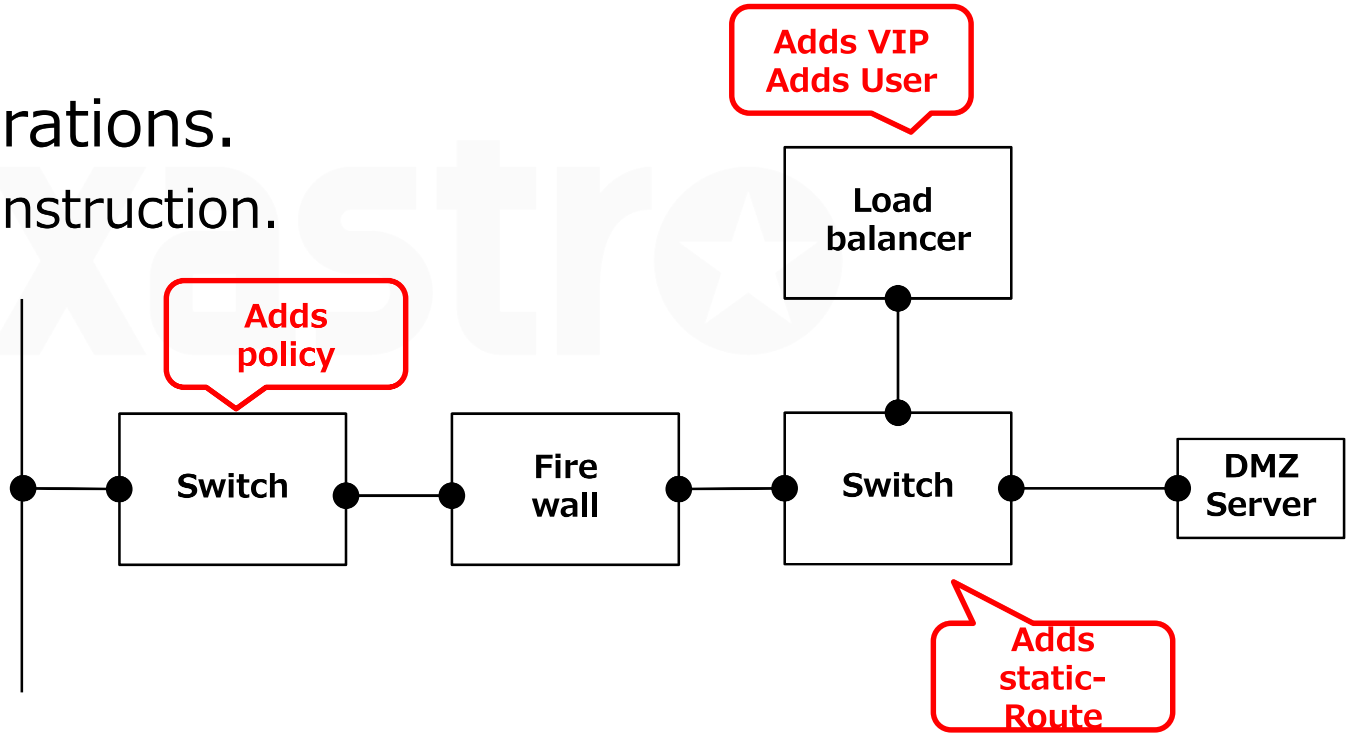
- Adding more network devices in a carrier type system
- Automate the operations of adding virtual IP and compare the operational costs of with and without automation.

Construction of the automated operations.

- Refer to the picture on the left for the construction.
- Total of 30 network devices

Automation construction and tasks

- Add Virtual IP and Member to Load balancer.
- Add policy to firewall
- Add static-route to switch.



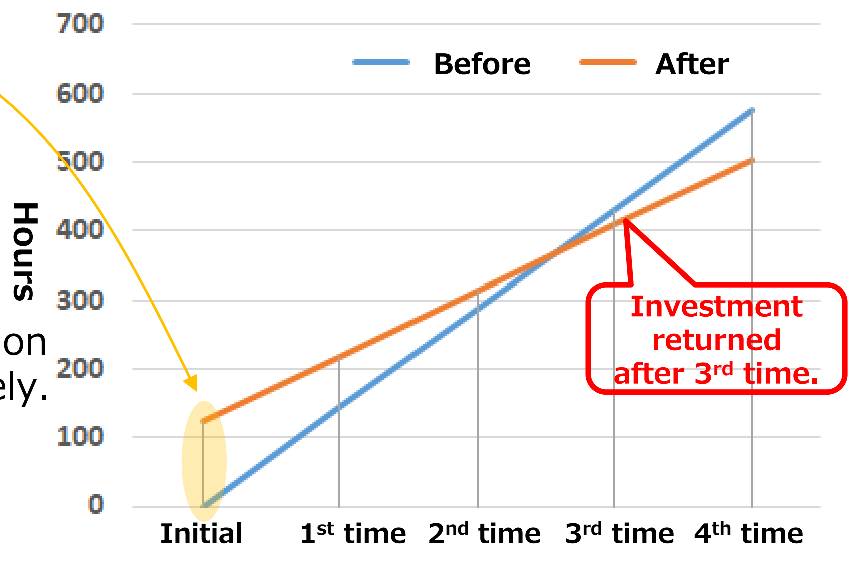
Case: Constructing Network Device(2/2)

Increase/Decrease in man-hours before and after automation + added work.

		Defining	Basic Design	Detailed Design	Operation design	Production	Evaluation		Total
Before	Hours(Per worker)	20.1	22.4	11.2	0	19.7	12	58.4	143.8
After	Hours(Per worker)	28.7	20.6	20.3	0	12.1	4	9.5	95.2
	Increase/Decrease(%)	(↑43%)	(↓8%)	(↑81%)	-----	(↓39%)	(↓67%)	(↓84%)	(↓34%)
	Added work	Consider Automation		Register CMDB		Run Jobflow			

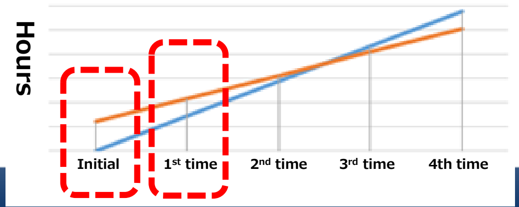
Return on Investment Concept.

- Man-hours used for Automation (Initial) : 123.4H
  - Step 1 : 44.7H　Step 2 : 63.5H　Step 3 : 15.2H
- Hours before Automation : 143.8H ⇒ After Automation : 95.2H
  - The number of man hours is reduced by34%. Additionally, the investment returns profit after the Third time (including the Initial stage)
- Depending on the case, preparation for automation and implementing the automation may be done separately or at the same time. In this case, they were done separately.

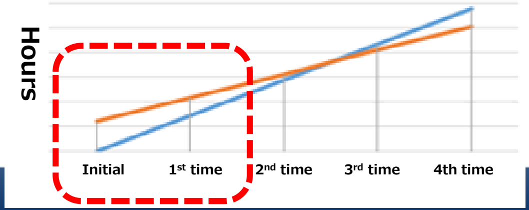


Graph of Man-hours (costs)

Individually implemented



Implemented at the same time



- Implementing automated SI
  - Effects and Estimations
  - Post-Automation Process changes and results.

Exastro

Defining Requirements

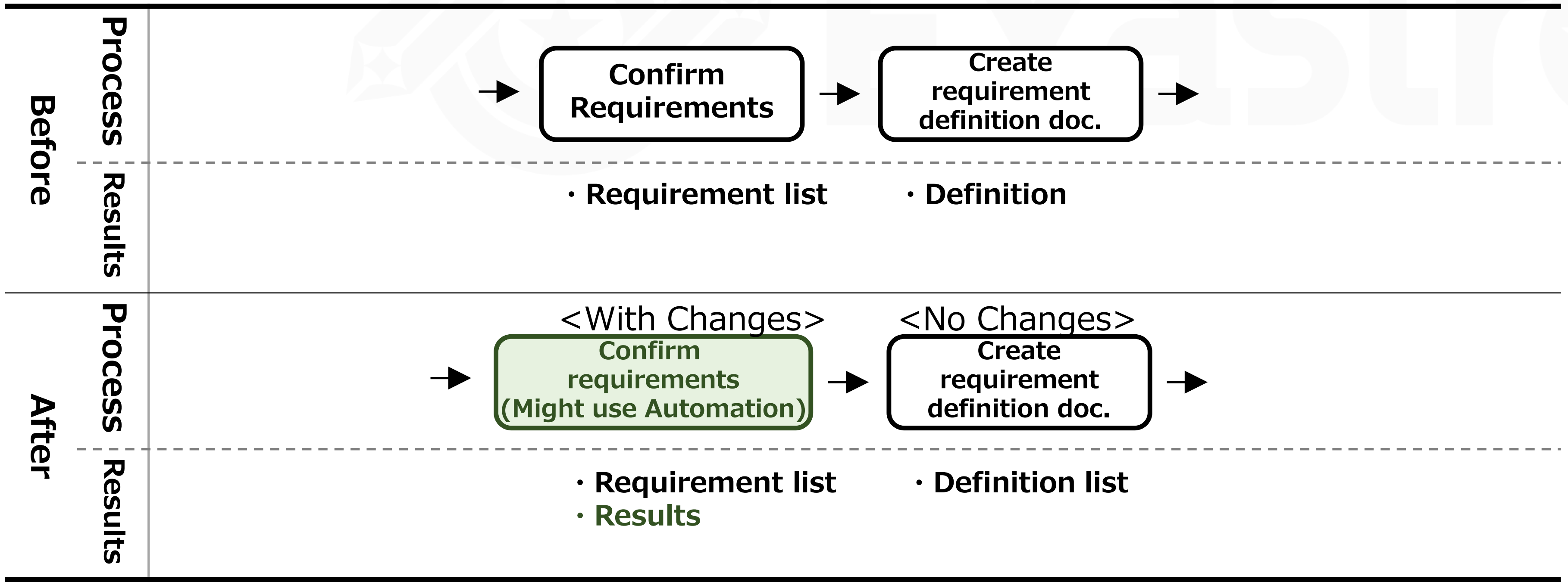
Changes in QCD per phase

Legend: ☺ No changes 😄 Better 😓 Might have additional work

	Defining			Design			Det.Design			Op. Design			Production			Test			Release		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
Before	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
After	☺	😓	😓	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄

Product and Process changes

Legend : No changes ⬜ Work With changes 🟩 Work Add 🟥 Work De-lete ⬜ Work



Explanation

At the defining stage, the scope of where Automation should be applied, etc. needs to be discussed and agreed upon. Therefore, C and D will increase.

Design

Changes in QCD per phase

Legend: ☺ No changes 😄 Better 😓 Might have additional work

	Defining			Design			Det.Design			Op. Design			Production			Test			Release		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
Before	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
After	☺	😓	😓	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄

Product and Process changes

Legend : No changes **Work** With changes **Work** Add **Work** De-lete **Work**

Before	**Process**	→ **Create Design Document** →
	Results	・**Design Document**
After	**Process**	<No changes> → **Create Design Document** →
	Results	・**Design Document**

Explanation

Since the contents that are going to get incorporated into the Design phase already is decided in the preparation phase, there is no work to be added here.

Det. Design

Changes in QCD per phase

Legend: ☺ No changes 😄 Better 😓 Might have additional work

	Defining			Design			Det.Design			Op. Design			Production			Test			Release		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
Before	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
After	☺	😓	😓	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄

Product and Process changes

Legend : No changes **Work** With changes **Work** Add **Work** De-lete **Work**

Before	**Process**	**Parameter design** →	**Create Parameter sheet** →	**Create operation procedure** →
	Results	· **Parameter design document**	· **Parameter sheet**	· **Operation procedure**
After	**Process**	<No changes> **Parameter design** →	<With Changes> **Register parameter to CMDB** →	<Delete> Create operation procedure →
	Results	· **Parameter design document**	· **CMDB**	· Operation procedure

Explanation

Parameters created in the parameter design will be registered to the CMDB. This will formalize parameters and help eliminate ambiguity, improving Q.

Additionally, the operation procedures, such as the order of application of parameters, will be replaced by the job flow created in the early preparation stage. As a result , creating operation procedures will be deleted. This will improve both C and D

Op.Design

Changes in QCD per phase

Legend: 🙂 No changes 😄 Better 😓 Might have additional work

	Defining			Design			Det.Design			Op. Design			Production			Test			Release		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
Before	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂	🙂
After	🙂	😓	😓	🙂	🙂	🙂	😄	😄	😄	🙂	🙂	🙂	😄	😄	😄	🙂	🙂	🙂	😄	😄	😄

Product and Process changes

Legend : No changes **Work** With changes 🟩 **Work** Add 🟥 **Work** De-lete **Work**

Before	**Process**	→ **Create monitoring design document** →	**Create Operation procedures** →
	Results	**· Monitoring design document**	**· Operation Procedures**
After	**Process**	<No changes> → **Create monitoring design document** →	<No changes> **Create Operation procedures** →
	Results	**· Monitoring design document**	**· Operation Procedures**

Explanation

Since this section focuses on automating construction, automating the operations is not taken into consideration.

When operational automation is implemented, the process and QCD will most likely change.

Production

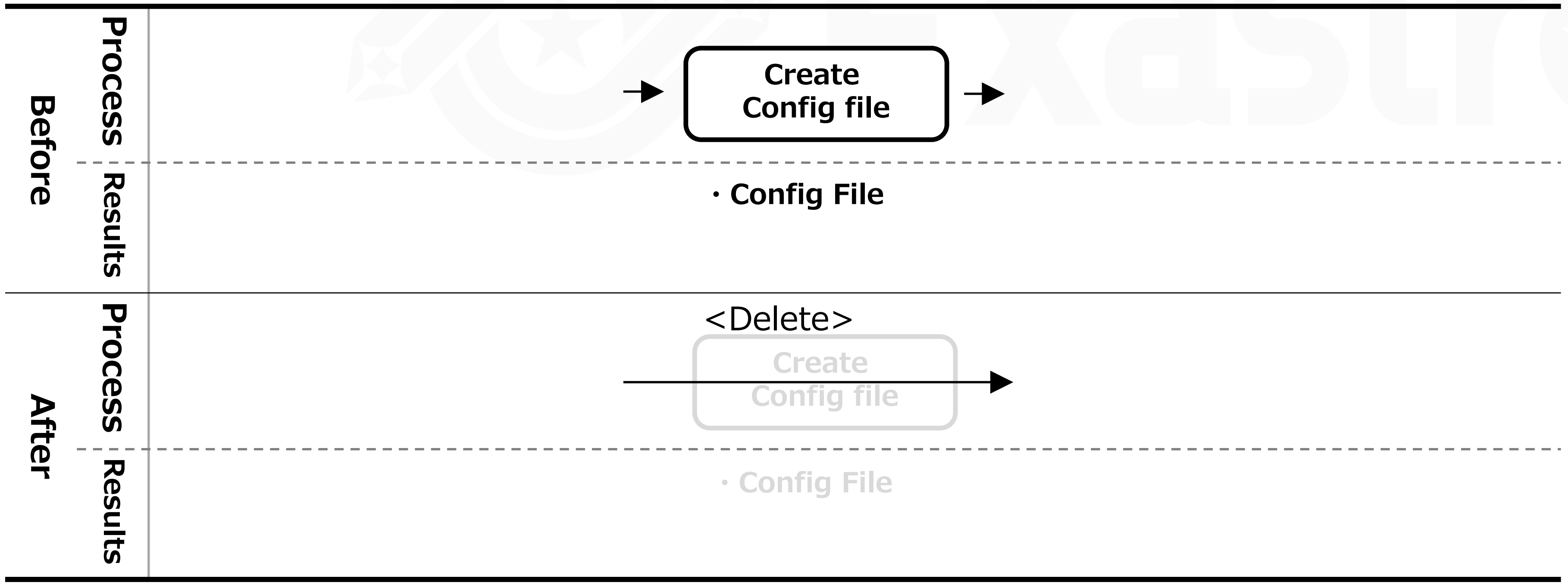
Changes in QCD per phase

Legend: ☺ No changes 😄 Better 😓 Might have additional work

	Defining			Design			Det.Design			Op. Design			Production			Test			Release		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
Before	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
After	☺	😓	😓	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄

Product and Process changes

Legend : No changes **Work** With changes **Work** Add **Work** De-lete **Work**



Explanation

The configuration file is created based on the Detailed design. It is automatically generated from IaC and CMDB, so the tasks of creating config files is deleted.

Test

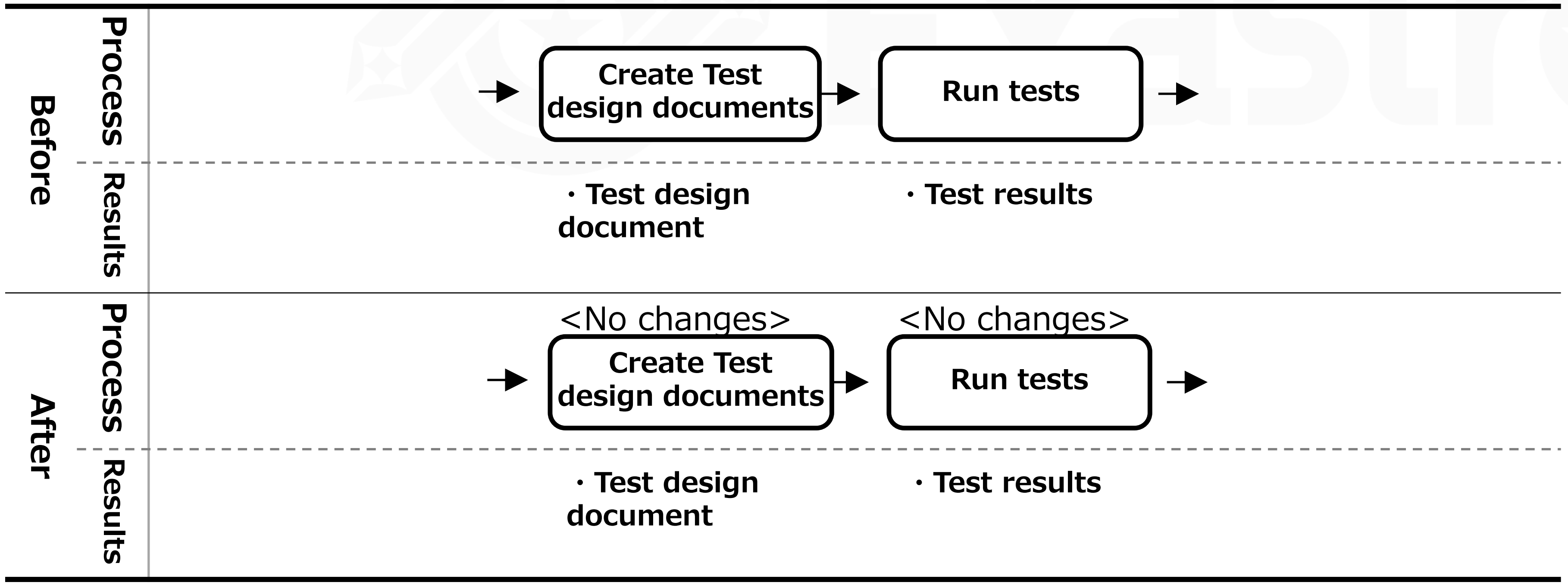
Changes in QCD per phase

Legend: ☺ No changes 😄 Better 😓 Might have additional work

	Defining			Design			Det.Design			Op. Design			Production			Test			Release		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
Before	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
After	☺	😓	😓	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄

Product and Process changes

Legend : No changes ⬜ Work With changes 🟩 Work Add 🟥 Work De-lete ▨ Work



Explanation

Again, this time, we're focusing on automating the construction of a system. Therefore, the test itself is not getting automated.

Similar to the production phase, the QCD/process will change if the test phase is automated.

Release

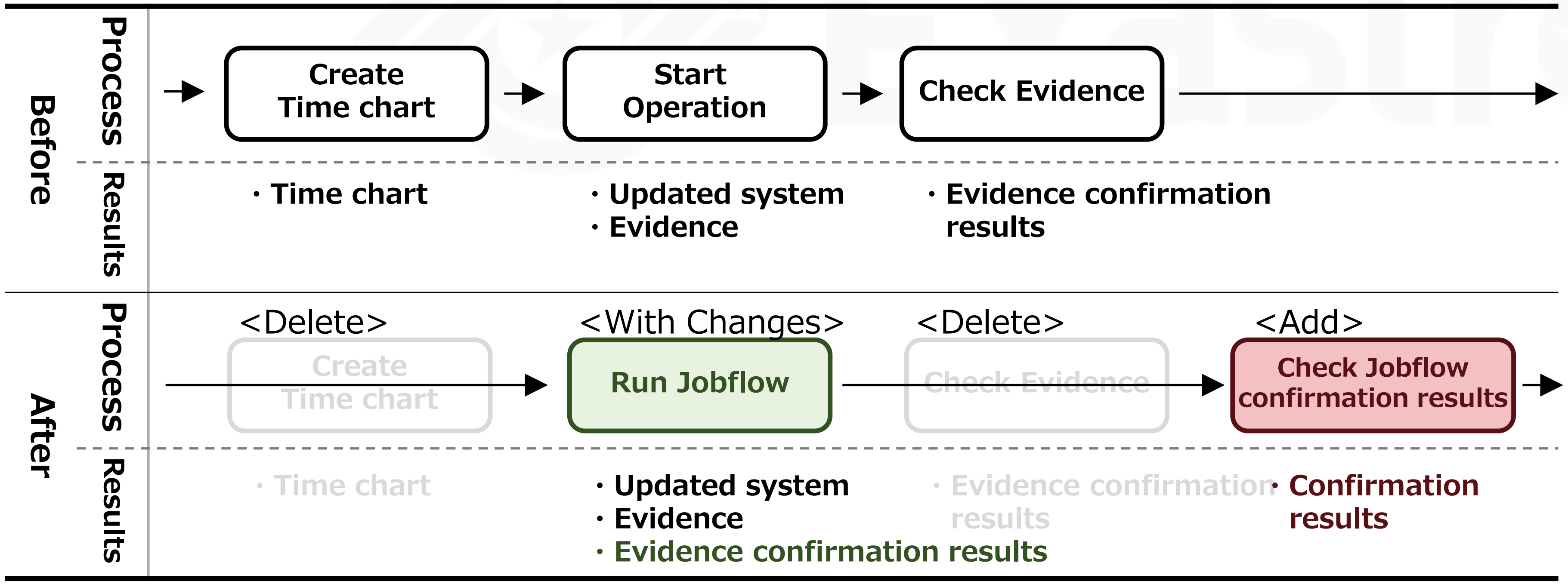
Changes in QCD per phase

Legend: ☺ No changes 😄 Better 😓 Might have additional work

	Defining			Design			Det.Design			Op. Design			Production			Test			Release		
	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D	Q	C	D
Before	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
After	☺	😓	😓	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄	☺	☺	☺	😄	😄	😄

Product and Process changes

Legend : No changes ⬜ Work With changes 🟩 Work Add 🟥 Work De-lete ⬜ Work



Explanation

The main task is to run the job flow created in the Detailed Design phase.

Since the time chart is replaced by the Jobflow, it will be deleted.

Since the evidence is checked in the Jobflow, the evidence check task will also be deleted.

Therefore, execution of the job

Summary

Summary

By following step 1-3, we can automate system operation/construction. Additionally, by changing the process, we can improve the efficiency of the automation.

TO-BE

Automated system construction/operation

Preparing for Automation (Step 1、Step 2、Step 3)

➕

Implementing Automated SI(Changes to the process and results)

AS-IS

Manual system construction/operation

Exastro