JP1 Version12

# JP1/Automatic Job Management System 3
# JP1/Client Process Automation
# ×
# Cloud 版 UiPath Orchestrator 連携ガイド
# （第 2 版）

2022 年 11 月

株式会社 日立製作所

マネージドサービス事業部
クラウドマネージドサービス本部
JP1-RPA 支援センター

# はじめに

本資料は，「JP1/Automatic Job Management System 3」および 「JP1/Client Process Automation」と「UiPath Automation Cloud for enterprise」で提供されるサービス「UiPath Orchestrator」との連携方法を明らかにすることで，業務自動化の範囲を広げ，運用の効率向上を図ることを目的としています。

■対象読者

本資料は，次の知識をお持ちの方にお読みいただくことを前提としています。

・UiPath に関する基本的な機能や操作

・Windows のバッチプログラムの作成

■お願い事項

本資料に記載の情報は，2022 年 11 月時点の特定環境における連携の検証結果を，事例としてご紹介するものです。

・可能な限り正確な内容を記載するよう努めていますが，その内容を保証するものではありません。

・記載された内容によって生じた損害等の一切の責任を負いかねますので，ご了承ください。

・画面表示をはじめ，製品仕様は，改良のため変更することがあります。

ご利用に際しては，お客さま，パートナー様ご自身でご検証のうえ，ご判断いただけますようお願いいたします。

■輸出時の注意

本製品を輸出される場合には，外国為替及び外国貿易法の規制ならびに米国の輸出管理規則など外国の輸出関連法規をご確認のうえ，必要な手続きをお取りください。

なお，ご不明な場合は，当社担当営業にお問い合わせください。

■発行

2020 年　10 月（初版）

2022 年　11 月（第 2 版）



■変更内容

第 2 版での変更内容

| # | 追加・変更内容 | 変更箇所 |
|---|---|---|
| 1 | 実行するプロセスおよびロボットを管理しているフォルダ名を指定するオプションを追加 | 6.1、7.2.1、8.1.4、8.1.6、8.1.7 |
| 2 | TSL プロトコル 許可していない場合の定義追加 | 8.1.6、8.1.7 |

なお、単なる誤字・脱字などは、お断りなく訂正しました

# 目次

1.　製品の総称および略称

本資料で使用する製品・サービスの総称および略称を次に示します。

**表 1.1 製品・サービス 総称**

| 総称 | 説明 |
|---|---|
| JP1/Automatic Job Management System 3 | RPA 製品が実行する業務を含め、サーバで行う基幹業務の実行スケジュールや実行順序を制御する製品 |
| JP1/Client Process Automation | RPA 製品が実行する業務を含め、PC で行う定型業務の実行スケジュールや実行順序を制御する製品 |
| UiPath Orchestrator | UiPath 社が運営する「UiPath Automation Cloud for enterprise」で提供される運用・管理サービス |
| UiPath プロセス実行スクリプト | UiPath のプロセス（ジョブ）を実行するスクリプト。詳細は、「8.1 UiPath プロセス実行スクリプト」を参照ください。 |

**表 1.2 製品・サービス 略称**

| 略称 | 正式名称 |
|---|---|
| JP1/AJS3 | JP1/Automatic Job Management System 3 |
| JP1/AJS3 - Manager | JP1/Automatic Job Management System 3 - Manager |
| JP1/AJS3 - View | JP1/Automatic Job Management System 3 - View |
| JP1/CPA | JP1/Client Process Automation |
| UiPath OC | UiPath Orchestrator |

**表 1.3 マイクロソフト製品の表記**

| 表記 | 正式名称 |
|---|---|
| Windows 10 | Windows(R) 10 Pro<br>Windows(R) 10 Enterprise |
| Windows Server | Microsoft(R) Windows Server(R) |
| Windows Server 2016 | Microsoft(R) Windows Server(R) 2016 Datacenter<br>Microsoft(R) Windows Server(R) 2016 Standard |
| PowerShell | Windows PowerShell(R) |

## 2. 前提プログラム

本資料の作成にあたり使用したプログラムおよびバージョンを次に示します。

**表 2.1 前提プログラム一覧**

| 対象ホスト・サービス | 前提プログラム | |
|---|---|---|
| マネージャーホスト | JP1/AJS3 - Manager | 12-00 |
| | JP1/AJS3 - View | 12-00 |
| | JP1/Base | 12-00 |
| 運用管理・ロボット実行端末 | JP1/CPA | 12-00 |
| | UiPath UnAttended Robot | 18.4.5 |
| UiPath Automation Cloud for enterprise | UiPath OC | |

## 3. 関連ドキュメント

関連するドキュメントを次に示します。

**表 3.1 関連ドキュメント**

| 製品・サービス名 | ドキュメント名 |
|---|---|
| JP1/AJS3 | JP1 Version 12 JP1/Automatic Job Management System 3 操作ガイド |
| JP1/CPA | JP1 Version 12 JP1/Client Process Automation |
| | ジョブ管理 基本ガイド（クライアント業務自動化編） |
| UiPath OC | UiPath Orchestrator ガイド |

4.　システム構成

システムの構成例を次に示します。

（１）JP1/AJS3 連携システム構成

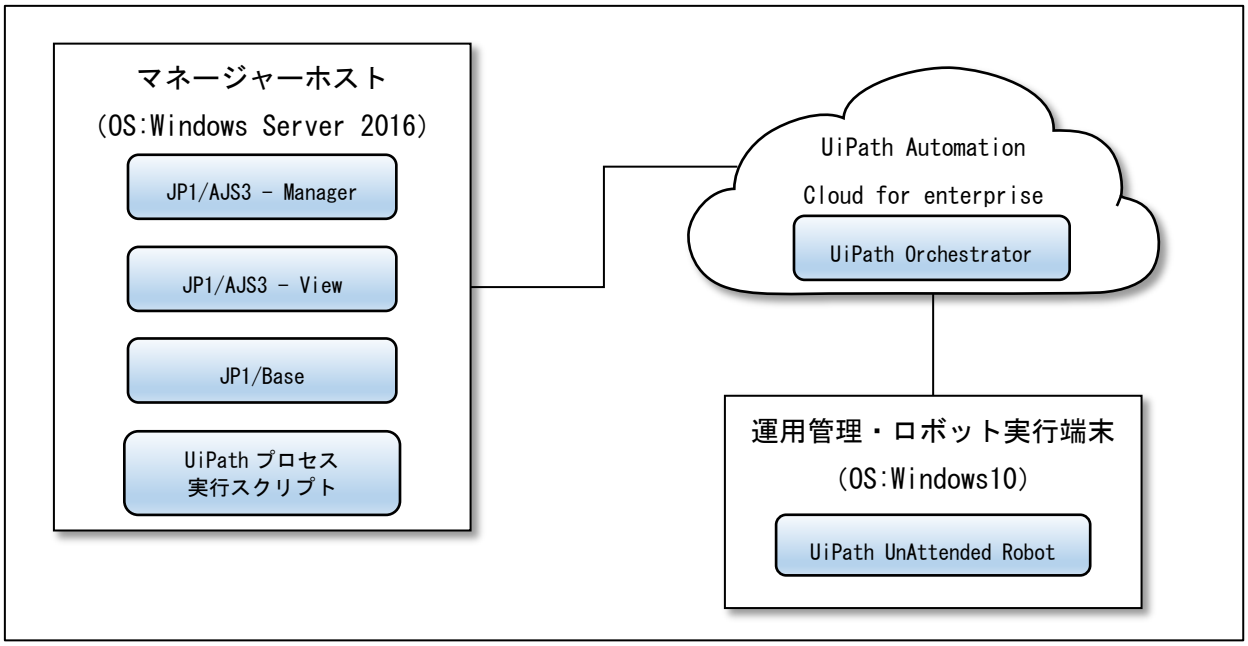



**図 4.1 JP1/AJS3 連携システム構成例**

（２）JP1/CPA 連携システム構成

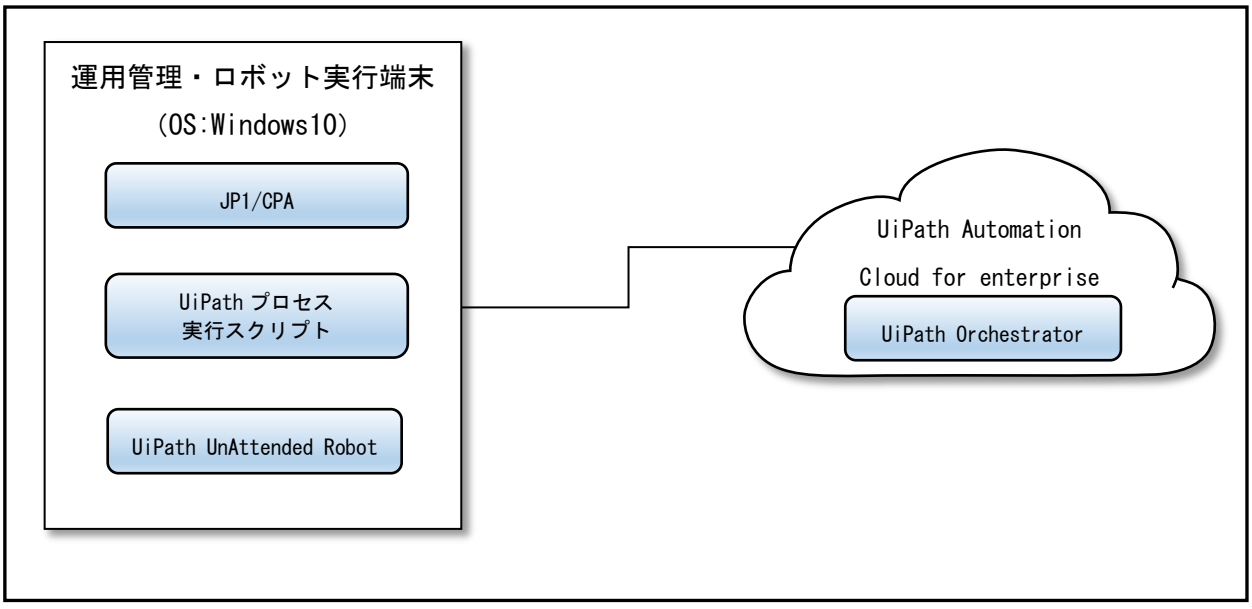
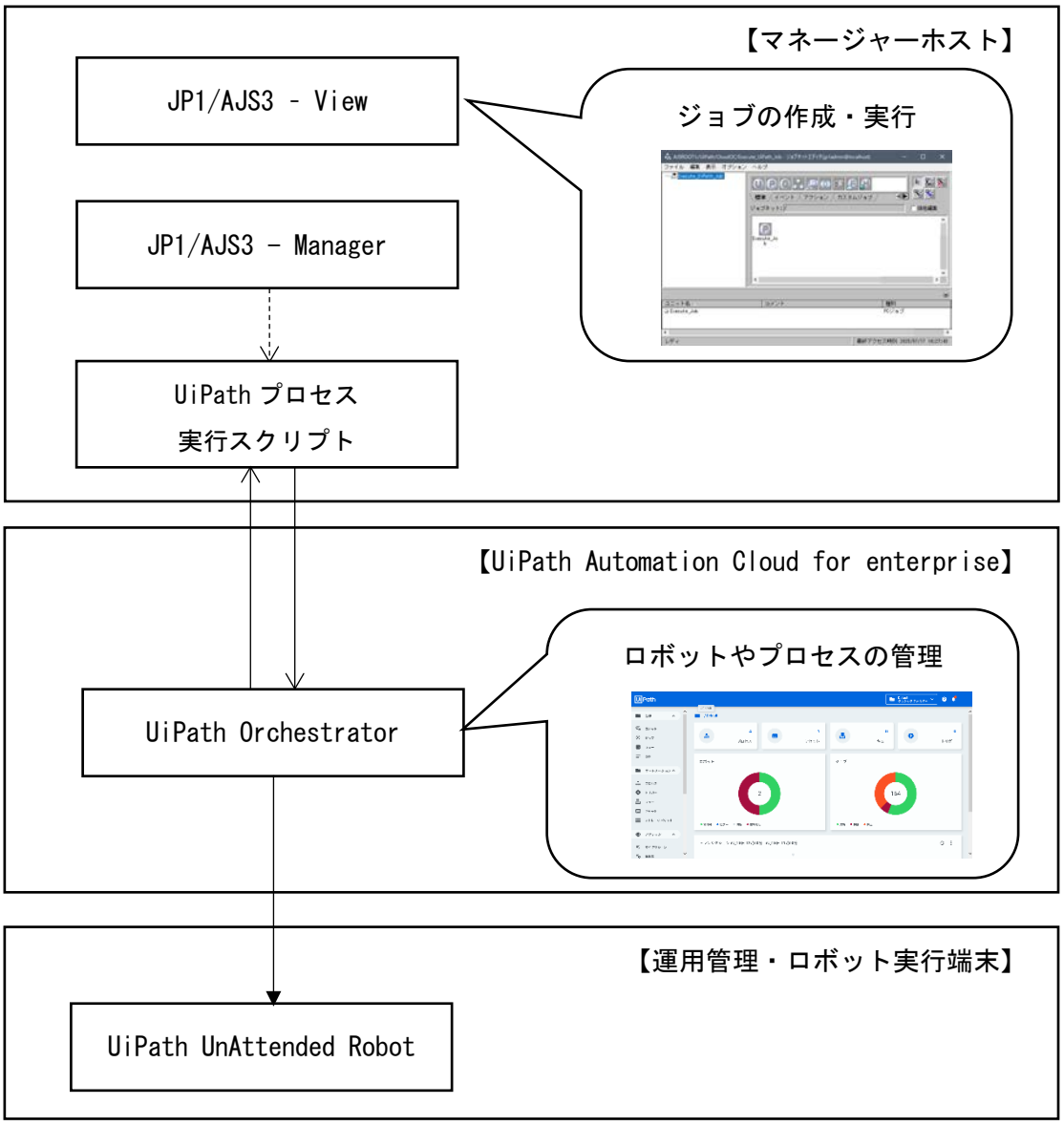



**図 4.2 JP1/CPA 連携システム構成例**

# 5.　概要

## 5.1.　JP1/AJS3 と UiPath OC との連携

JP1/AJS3 のジョブで UiPath プロセス実行スクリプトを実行することで、ロボット実行端末に定義されている UiPath のプロセスまたは、ワークフローの順序制御や定期実行など実行スケジュールを管理することができます。




【凡例】

------≫ ：UiPath プロセス実行スクリプトの実行

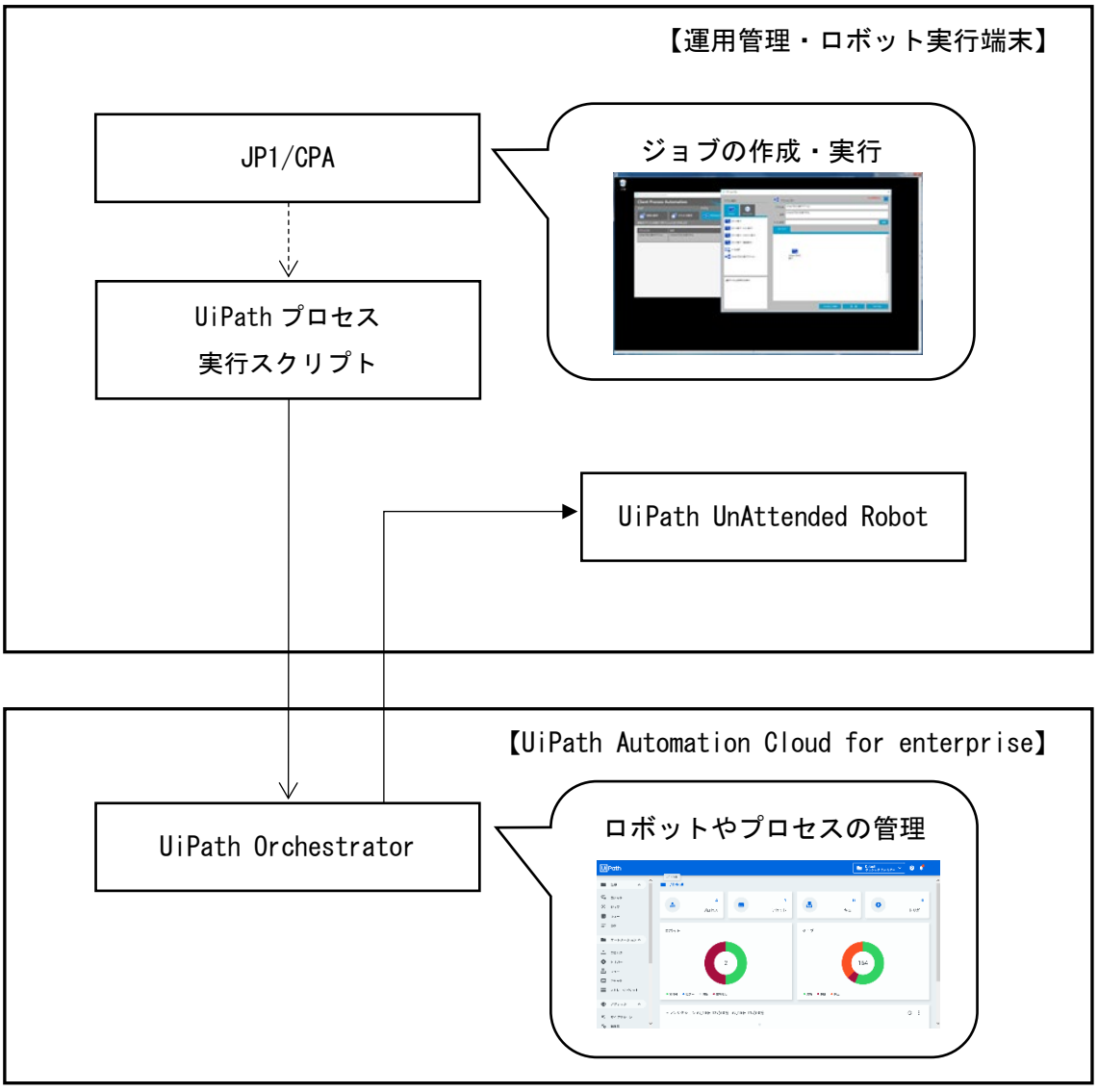
——≫ ：プロセスまたは、ワークフローの実行依頼または応答

——▶ ：プロセスまたは、ワークフローの実行

**図 5.1 概要図(1)**

## 5.2. JP1/CPA と UiPath OC との連携

運用管理端末に存在する JP1/CPA では、JP1/CPA のジョブとして UiPath プロセス実行スクリプトを登録することで、ロボット実行端末に定義されている UiPath のプロセスまたは、ワークフローの順序制御や定期実行など実行スケジュールを管理することができます。




【凡例】

------> ：UiPath プロセス実行スクリプトの実行

——> ：プロセスまたは、ワークフローの実行依頼

——▶ ：プロセスまたは、ワークフローの実行

**図 5.2 概要図(2)**

### 5.3. JP1/AJS3 と UiPath OC との連携に関する前提条件

① マネージャーホスト(JP1/AJS3 が導入されている端末)と、「UiPath Automation Cloud for enterprise (UiPath Orchestrator)」の通信が可能であること。

② 運用管理・ロボット実行端末（JP1/CPA が導入されている端末 および UiPath UnAttended Robot が導入されている端末）と、「UiPath Automation Cloud for enterprise (UiPath Orchestrator)」の通信が可能であること。

③ プロキシ環境下で UiPath UnAttended Robot を利用する場合、以下の HP に掲載された設定を行うこと。
https://www.uipath.com/ja/resources/knowledge-base/settings-for-proxy-environments

### 5.4. JP1/CPA と UiPath OC との連携に関する前提条件

① 運用管理・ロボット実行端末（JP1/CPA が導入されている端末 および UiPath UnAttended Robot が導入されている端末）と、「UiPath Automation Cloud for enterprise (UiPath Orchestrator)」の通信が可能であること。

② プロキシ環境下で UiPath UnAttended Robot を利用する場合、以下の HP に掲載された設定を行うこと。
https://www.uipath.com/ja/resources/knowledge-base/settings-for-proxy-environments

# 6. JP1 と UiPath OC の連携方法

## 6.1. JP1/AJS3 と UiPath OC の連携方法

JP1/AJS3 - View の[ジョブネットエディタ]ウィンドウで、UiPath プロセス実行スクリプトを実行するジョブを作成して実行登録してください。

ジョブの定義で指定する UiPath プロセス実行スクリプト(※1)の「実行形式」、「実行ファイル」、「オプション」について以下に示します。

※1：UiPath プロセス実行スクリプトについては、「8.1 UiPath プロセス実行スクリプト」を参照ください。

● 実行形式

```
UiPath プロセス実行スクリプトのインストールフォルダ¥orchestrator-job.bat
-p プロセス名 [-i カスタム値] [-r 実行ロボット名] [-o フォルダ名]
```

● 実行ファイル

UiPath プロセス実行スクリプトのインストールフォルダに存在する「orchestrator-job.bat」ファイルをフルパスで指定します。

● オプション

-p (-process)

実行するプロセス名を指定します。指定できるプロセス名は UiPath Orchestrator「オートメーション」メニューの「プロセス」で確認できます。


**図 6.1 プロセス名 表示例**

-i (-input) カスタム値

ワークフローを実行するときに，カスタム値（※1）を JSON 形式の In または In / Out 引数に割り当てることができます。 入力引数として文字列の配列と配列を使用できます。

※1：カスタム値の詳細については、以下の URL を参照ください。
https://docs.uipath.com/orchestrator/lang-ja/docs/about-input-and-output-parameters#
指定されたカスタム値は、Orchestrator API のジョブレベル -/odata/Jobs/UiPath.Server.Configuration.Odata.StartJob エンドポイントへの POST 要求として入力します。上記 Orchestrator API の詳細については以下の URL を参照してください。
https://docs.uipath.com/orchestrator/lang-ja/v2019/reference/jobs-requests

-r（-robotname）実行ロボット名
実行するロボット名を指定することができます。指定できるロボット名は UiPath Orchestrator「管理」メニューの「ロボット」で確認できます。
指定しない場合は、構成定義に記載したロボット名が連携されます。ロボット名の定義に関しては、「UiPath プロセス実行スクリプト 取扱説明書」を参照してください。

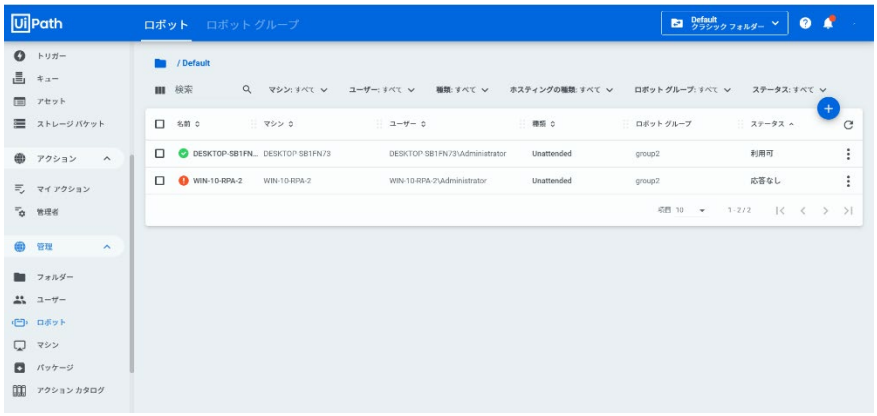



**図 6.2 ロボット名 表示例**

-o（-organizationunit）フォルダ名
実行するプロセスおよびロボットを管理しているフォルダ名（※2）を指定することができます。指定できるフォルダ名は UiPath Orchestrator「テナント」メニューの「フォルダ」で確認できます。
指定しない場合は、構成定義に記載したフォルダ名が連携されます。
構成定義にフォルダ名を記載し、かつ、オプションでフォルダ名を指定した場合は、オプションで指定したフォルダ名が連携されます。
構成定義にフォルダ名を記載せず、かつ、オプションも指定しない場合は、フォ

ルダ指定なし（※3）で動作します。

※2：指定可能はフォルダの種類は、クラシックフォルダーのみです。
モダンフォルダーの指定には対応しておりません。

※3：構成定義に記載したテナントで管理するフォルダが 1 つのみの場合は、フォルダ指定なしでプロセスの実行が可能です。
テナントで管理するフォルダが複数ある場合は、必ず構成定義またはオプションにてフォルダ名を指定して下さい。

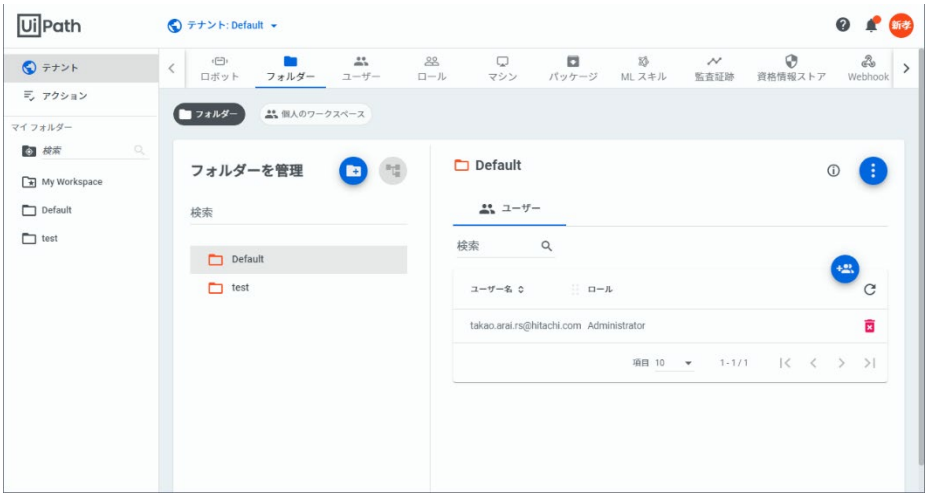



**図 6.3 フォルダ名 表示例**

※オプションの指定例

```
orchestrator-job.bat -p Sample
orchestrator-job.bat -p Sample -i ¥"{'arrayInt':['1','2','3']}¥"
orchestrator-job.bat -p Sample -r SampleRobot
orchestrator-job.bat -p Sample -r SampleRobot -o SampleFolder
orchestrator-job.bat -p Sample -i ¥" {‘arrayInt’:[ ‘1’ ,’ 2’ ,’ 3’ ]}¥”  -r SampleRobot
```

## 6.2. JP1/CPA と UiPath OC の連携方法

JP1/CPA のアクションで UiPath プロセス実行スクリプトを実行してください。

UiPath プロセス実行スクリプトの「実行形式」、「実行ファイル」、「オプション」については、「6.1 JP1/AJS3 と UiPath OC の連携方法」を参照してください。

## 7. JP1/CPA における「UiPath UnAttended Robot」の実行方法の切り替え

JP1/CPA での従来の実行方法（※1）から、「UiPath OC」経由の実行方法に切り替える際の前提条件と、切り替え手順を以下に示します。

※1：「UiPath UnAttended Robot」を直接実行する方法

### 7.1. 前提条件

・「UiPath OC」へ API の発行ができること

・「UiPath OC」にプロセス、実行対象ロボットが登録されていること

・「UiPath OC」から実行対象ロボットに実行指示が可能であること

### 7.2. 切り替え手順

実行方法の切り替えに関する以下の手順を示します。

・JP1/CPA ユニット定義の変更

#### 7.2.1. JP1/CPA ユニット定義の変更

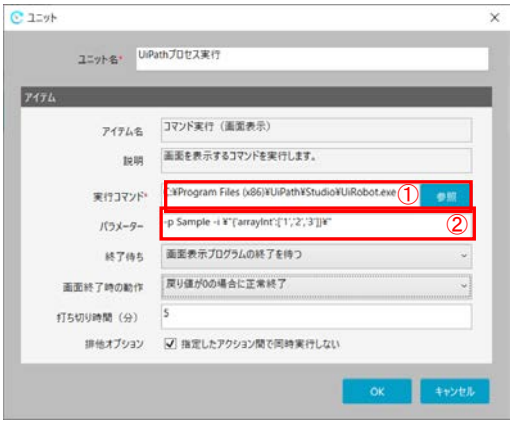
既に定義されているユニットについて、以下の項目を変更する必要があります。

・ 実行コマンド（①）

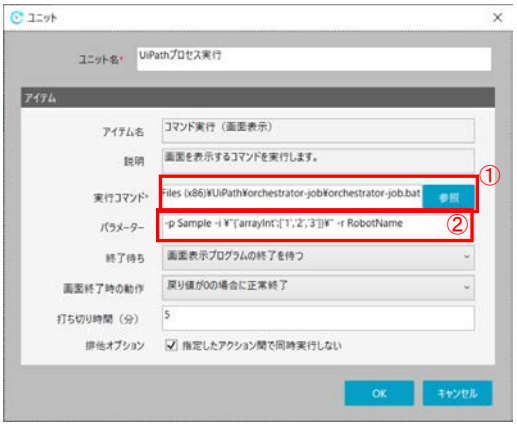
・ パラメーター（②）

変更内容について、以下に示します。

【変更前 ユニット設定】



【変更後 ユニット設定】




① 「UiPathRobot インストールフォルダ¥UiRobot.exe」から
「UiPath プロセス実行スクリプトインストールフォルダ¥orchestrator-job.bat」
に変更してください。

② 基本的にパラメーターの変更は不要ですが、アクション/ジョブ毎にロボットを変更する場合は、以下#5 のパラメーター（実行ロボット名）を指定する必要があります。

**表 7.1 UiPath プロセス実行スクリプトパラメーター一覧**

| # | パラメーター | 説明 | 必須/任意（※1） | 備考 |
|---|---|---|---|---|
| 1 | -p(-proccess) | 実行プロセス名 | ◎ | |
| 2 | -f(-file) | ワークフローファイル名 | △ | 指定されていた場合は、無視されます。 |
| 3 | -i(-input) | カスタム値 | ○ | |
| 4 | --rdp | RDP の有無 | △ | 指定されていた場合は、無視されます。 |
| 5 | -r(-robotname) | 実行ロボット名 | ○ | 本パラメーターにてプロセスを実行するロボットを指定できます。本パラメーターの指定を省略した場合、UiPath プロセス実行スクリプトの構成ファイル「config.ps1」に設定したロボットでプロセスを実行します。 |

| | | | | |
|---|---|---|---|---|
| 6 | -o(-organizationunit) | フォルダ名 | ○ | 本パラメーターにて、実行するプロセスとロボットを管理するフォルダを指定できます。<br>本パラメーターの指定を省略した場合、UiPath プロセス実行スクリプトの構成ファイル「config.ps1」に設定したフォルダでプロセスを実行します。 |

※1：◎：必須、○：任意、△：任意（無視）、－：指定不可

# 8. 付録

## 8.1. UiPath プロセス実行スクリプト

### 8.1.1. 前提 OS

本スクリプトの前提となる OS を以下に示します。

**表 8.1 UiPath プロセス実行スクリプトの前提 OS**

| 製品名 | バージョン |
|---|---|
| Windows | 8 以降 |
| Windows Server | 2012 以降 |

### 8.1.2. 前提プログラム

本スクリプトの前提プログラムを以下に示します。

**表 8.2 UiPath プロセス実行スクリプトの前提プログラム**

| 製品名 | バージョン |
|---|---|
| PowerShell | 3 以降 |

### 8.1.3. ファイル構成

本スクリプトのファイルの構成を以下に示します。

**表 8.3 UiPath プロセス実行スクリプトのファイル構成**

| # | パス | 説明 | 備考 |
|---|---|---|---|
| 1 | インストールフォルダ | - | |
| 2 | +- orchestrator-job.bat | UiPath プロセス実行スクリプトファイル | |
| 3 | +- orchestrator-job.ps1 | 内部プログラムファイル | |
| 4 | +- config.ps1 | 設定ファイル | |

### 8.1.4. スクリプト引数

```
orchestrator-job.bat -p (-process) 実行プロセス名
                     [-i (-input) カスタム値]
                     [-r (-robotname) 実行ロボット名]
                     [-o (-organizationunit) フォルダ名]
```

**表 8.4 UiPath プロセス実行スクリプトのコマンド引数**

| # | 引数 | 説明 | 必須 | 備考 |
|---|---|---|---|---|
| 1 | -p (-proccess) | 実行プロセス名<br>例：<br>orchestrator-job.bat -p ProcessName<br>orchestrator-job.bat -process ProcessName | ○ | 未入力の場合は、異常終了となります。 |
| 2 | -i (-input) | カスタム値<br>例：<br>orchestrator-job.bat -p ProcessName -i ¥"{'arrayInt':['1','2','3']}¥"<br>orchestrator-job.bat -process ProcessName -input ¥"{'arrayInt':['1','2','3']}¥" | × | |
| 3 | -r (-robotname) | 実行ロボット名<br>例：<br>orchestrator-job.bat -p ProcessName -r RobotName<br>orchestrator-job.bat -process ProcessName -robotname RobotName | × | |
| 4 | -o (organizationunit) | フォルダ名<br>例：<br>orchestrator-job.bat -p ProcessName -o FolderName<br>orchestrator-job.bat -p ProcessName -organizationunit FolderName | × | |

上記以外の引数が指定された場合は、無視されます。

### 8.1.5. スクリプト終了コード

0：正常終了

1：異常終了

### 8.1.6. ユーザ可変値

以下のパラメーターは，設定ファイル（config.ps1）で定義されています。
「config.ps1」をエディタで開き、適宜変更します。

**表 8.5 UiPath プロセス実行スクリプトのユーザ可変値**

| # | パラメータ | 必須／任意 | 内容 |
|---|---|---|---|
| 1 | $C_Tenant ※1 | 必須 | UiPath で定義されているテナント名を指定します。 |
| 2 | $C_UserKey ※2 | 必須 | UiPath で定義されているユーザーキーを指定します。 |
| 3 | $C_LogicalAccountName ※2 | 必須 | UiPath で定義されているアカウントの論理名を指定します。 |
| 4 | $C_LogicalTenantName ※2 | 必須 | UiPath で定義されているテナントの論理名を指定します。 |
| 5 | $C_FolderNam | 必須 | UiPath で定義されているフォルダ名（クラシックフォルダーの名前）を指定します。<br>ただし、コマンド実行時の引数 -o または -organizationunit が指定されている場合は、引数で指定したフォルダ名が優先されます。 |
| 6 | $C_ClientId ※2 | 必須 | UiPath で定義されているクライアント ID を指定します。 |
| 7 | $C_RobotName | 必須 | UiPath で定義した実行対象となるロボット名を指定します。<br>ただし、コマンド実行時の引数 -r または -robotname が指定されている場合は、引数で指定したロボット名が優先されます。 |
| 8 | $C_ProxyEnable | 必須 | プロキシサーバ経由でジョブを実行する場合は"true"を指定します。<br>プロキシサーバ経由でない場合は"false"を指定します。<br>何も設定されていない場合 および "true"/"false"以外が設定されている場合は、"false"を設定した場合と同様の動作となります。 |

| | | | |
|---|---|---|---|
| 9 | $C_ProxyUrl ※3 | 任意 | プロキシサーバ経由でジョブを実行する場合に指定します。 |
| 10 | $C_ProxyUser | 任意 | |
| 11 | $C_ProxyPassword | 任意 | |
| 12 | [Net.ServicePointManager]::SeacurityProtocol=[Net.SecurityProtocolType]::Tls12 | 任意 | UiPath プロセス実行スクリプトを実行する環境で、TSL プロトコルのバージョン1.2を許可していない場合にコメントを外して指定します。<br>コメントを外すには、先頭文字「#」を削除してください。 |

※1：設定する値は、Cloud Platform アカウントにログインし、サービスページで確認できます。

※2：設定する値は、Cloud Platform アカウントにログインし、サービスページから「API アクセス (API Access)」を選択することで確認できます。

※3：プロキシの設定は以下の内容で指定します。

*<プロトコル>://<proxy サーバのホスト名または IP アドレス>:<ポート番号>/*

### 8.1.7. スクリプト内容

（１）orchestrator-job.bat

```
@echo off

REM スクリプト実行
echo Start Script.
PowerShell -ExecutionPolicy RemoteSigned -File "%~dp0¥orchestrator-job.ps1" %*

REM 結果を出力
echo End Script. Return Code: %ERRORLEVEL%

REM 常に正常で終了
exit %ERRORLEVEL%

@echo on
```

（２）orchestrator-job.ps1

```
Param (
          [Alias("process")][string] $p = "",
          [Alias("input")][string] $i = "{}",
          [Alias("robotname")][string] $r = "",
          [Alias("organizationunit")][string] $o = ""
)

# Include config
$ScriptDir = Split-Path $MyInvocation.MyCommand.Path -Parent
. "$ScriptDir¥config.ps1"

# Constant
set-variable -name C_UriAuth -value "https://account.uipath.com/oauth/token" -option constant
set-variable -name C_UriOrch -value "https://platform.uipath.com" -option constant
```

```
set-variable -name C_ContentType -value "application/json;charset=utf-8" -option constant
set-variable -name C_HttpMethod_Get -value "Get" -option constant
set-variable -name C_HttpMethod_Post -value "Post" -option constant
set-variable -name C_Retry -value 20 -option constant
set-variable -name C_interval -value 3 -option constant

function WriteLog
{
          Param ($message, [switch] $err)

          $now = Get-Date -Format "G"
          $line = "$now`t$message"
          if ($err)
          {
                    Write-Error $line
          } else {
                    Write-Host $line
          }
}

function HttpRequest
{
    Param ($uri, $method, $headers, $body, $credntial)

    if ($C_ProxyEnable -eq "true") {
          $response = Invoke-RestMethod -Uri $uri -Proxy $C_ProxyUrl -ProxyCredential $credntial -
Method $method -Body $body -ContentType $C_ContentType -Headers $headers
    } else {
          $response = Invoke-RestMethod -Uri $uri -Method $method -Body $body -ContentType
$C_ContentType -Headers $headers
    }
    return $response
}

# Main
    if ([string]::IsNullOrEmpty($p)) {
          WriteLog "Parameter '-process' or '-p' is required." -err
          exit 1
    } else {
                    $execProc = $p
                    $customValue = $i
                    $robotName = if ([string]::IsNullOrEmpty($r)) { Write-Output $C_RobotName }
else { Write-Output $r }
                    $folderName = if ([string]::IsNullOrEmpty($o)) { Write-Output $C_FolderName }
else { Write-Output $o }
          }

          $scriptPath = Split-Path -Parent $MyInvocation.MyCommand.Path

# Proxy
          if ($C_ProxyEnable -eq "true") {
                    $securePassword = ConvertTo-SecureString $C_ProxyPassword -AsPlainText -Force
                    $proxyCredntial = New-Object System.Management.Automation.PSCredential -
ArgumentList $C_ProxyUser, $securePassword
          }

# Orchestrator Login
          $requestHeaders = @{}
          $bodyAccount = @{
```

```
                    "grant_type" = "refresh_token"
                    "client_id" = "$C_ClientId"
                    "refresh_token" = "$C_UserKey"
          } | ConvertTo-Json
          $resAccount = HttpRequest $C_UriAuth $C_HttpMethod_Post $requestHeaders $bodyAccount
$proxyCredntial

# Login check
          if ($resAccount -ne $null)
          {
                    $access_token = $resAccount.access_token
                    $id_token = $resAccount.id_token
                    $requestHeaders       =       @{"X-UiPath-TenantName"="$C_LogicalTenantName";
"Authorization"="Bearer $access_token"}
          } else {
                    WriteLog "ERROR: Orchestrator Login failed." -err
                    exit 1
          }

# Get folder id
          if ($folderName) {
                    $requestBody = @{}
                    $uriFolders                                                                    =
"$C_UriOrch/$C_LogicalAccountName/$C_Tenant/odata/Folders?`$filter=DisplayName eq '$folderName'"
                    $resFolders = HttpRequest $uriFolders $C_HttpMethod_Get $requestHeaders
$requestBody $proxyCredntial
                    if ($resFolders.value) {
                              $resFolders.value | ForEach-Object {
                                        $folderId = $_.Id
                              }
                              $requestHeaders.Add("X-UIPATH-OrganizationUnitId", $folderId)
                    } else {
                              WriteLog "ERROR: Folder doesn't exist. FolderName:[$folderName]" -
err
                              exit 1
                    }
          }

# Get robot info
          $requestBody = @{}
          $uriRobots = "$C_UriOrch/$C_LogicalAccountName/$C_Tenant/odata/Robots?`$filter=Name eq
'$robotName'"
          $resRobots = HttpRequest $uriRobots $C_HttpMethod_Get $requestHeaders $requestBody
$proxyCredntial
          $machineName = @()
          $robotEnvironments = @()
          $robotIds = @()
          if ($resRobots.value -ne $null) {
                    $resRobots.value | ForEach-Object {
                              $machineName += $_.MachineName
                              $robotEnvironments += $_.RobotEnvironments
                              $robotIds += $_.Id
                    }
          } else {
                    WriteLog "ERROR: Robot name doesn't exist. RobotName:[$robotName]" -err
                    exit 1
          }

# Get Processes
```

```
          $requestBody = @{}
          $uriProcess  =  "$C_UriOrch/$C_LogicalAccountName/$C_Tenant/odata/Releases?`$filter=Name
eq '$execProc'"
          $resProcess = HttpRequest $uriProcess $C_HttpMethod_Get $requestHeaders $requestBody
$proxyCredntial
          $resProcess.value | ForEach-Object {
                    $procId = $_.Id
                    $procKey = $_.Key
                    $procName = $_.Name
                    $procVer = $_.ProcessVersion
                    $procEnv = $_.EnvironmentName
          }

# Start Job
          if ($procKey)
          {
                    $uriStartJob                                                                    =
"$C_UriOrch/$C_LogicalAccountName/$C_Tenant/odata/Jobs/UiPath.Server.Configuration.OData.StartJobs
"
                    $bodyStartJob = @{
                              "startInfo" = @{
                                        "ReleaseKey" = $procKey
                                        "Strategy" = "Specific"
                                   "RobotIds" = $robotIds
                                        "JobsCount"= 0
                                        "InputArguments" = $customValue
                              }
                    } | ConvertTo-Json
                    # Execute Job
                    $resStartJob  =  HttpRequest  $uriStartJob  $C_HttpMethod_Post  $requestHeaders
$bodyStartJob $proxyCredntial
          } else {
                    WriteLog "ERROR: Process ID doesn't exist. ProcessName:[$execProc]" -err
                    exit 1
          }

# Check Job Status
          if ($resStartJob)
          {
                    $jobId = $resStartJob.value.Id
                    $uriJobState                                                                    =
"$C_UriOrch/$C_LogicalAccountName/$C_Tenant/odata/Jobs($jobId)"
                    $counter = 1
                    $requestBody = @{}
                    while ($counter -le $C_Retry)
                    {
                              # Check Job state
                              $resJobState   =   HttpRequest   $uriJobState   $C_HttpMethod_Get
$requestHeaders $requestBody $proxyCredntial
                              $jobState = $resJobState.State
                              # Break if Job is not pending or running
                              if ($jobState -ne "Pending" -and $jobState -ne "Running")
                              {
                                        $jobStartTime = $resJobState.StartTime
                                        $jobEndTime = $resJobState.EndTime
                                        $jobInfo = $resJobState.Info
                                        break
                              }
                              Start-Sleep -s $C_Interval
```

```
                $counter++
            }
            # Job result
            if ($jobState -eq "Successful")
            {
                # Successful (Exit Code=0)
                exit 0
            } else {
                # Unsuccessful (Exit Code=1)
                WriteLog "ERROR: Job is Unsuccessful. State:[$jobState]" -err
                $Env:errorlevel = 1
                exit 1
            }
    } else {
            # Start Job failure (Exit Code=1)
            WriteLog  "ERROR: Start Job failed. ProcessName:[$procName]" -err
            exit 1
    }
```

（３）config.ps1

```
# Orchestrator Service
# TenantName
$C_Tenant = ""

# API Settings
# UserKey
$C_UserKey = ""

# LogicalAccountName
$C_LogicalAccountName = ""

# LogicalTenantName
$C_LogicalTenantName = ""

# FolderName
$C_FolderName = ""

# ClientId
$C_ClientId = ""

# ExecRobotName
$C_RobotName = ""

# ProxySettings
$C_ProxyEnable = "false"
$C_ProxyUrl = ""
$C_ProxyUser = ""
$C_ProxyPassword = ""

# SecurityProtocolSettings
#[Net.ServicePointManager]::SecurityProtocol=[Net.SecurityProtocolType]::Tls12
```

－以上－