

# xStream Application Program Interface (API) Guide

Document Revision: XAPI4.15.0-01

Revision Date: September 7, 2020

**【重要】**

以下の記載内容ならびに画像は事前の予告なく変更の可能性があります。

# Table of Contents

<b>Table of Contents</b> .....	<b>2</b>
<b>1 xStream API</b> .....	<b>3</b>
1.1 概要.....	3
1.2 本ガイドの対象 .....	3
1.3 改訂履歴.....	3
1.4 商標について.....	4
<b>2 API リクエストの発行</b> .....	<b>5</b>
2.1 エンドポイント.....	5
2.2 APIの認証方法 .....	5
2.2.1 Python を用いた Authorization ヘッダーの生成例.....	5
2.2.2 PowerShellを用いた Authorization ヘッダーの生成例.....	7
2.3 Python を用いた API リクエストの発行例.....	9
<b>3 本サービスで利用可能なAPI一覧</b> .....	<b>15</b>
3.1 仮想マシン一覧の取得.....	15
3.2 仮想マシン情報の取得.....	15
3.3 仮想マシンのパフォーマンス情報の取得.....	16
3.4 仮想マシンの Power ON .....	17
3.5 仮想マシンの Power OFF.....	18
3.6 OS シャットダウン.....	18
3.7 ハードウェアテンプレート一覧の取得 .....	19
3.8 ハードウェアテンプレートの変更.....	19
3.9 キーペア失効日時の設定.....	20
3.10 タスク実行結果の取得 .....	21
<b>4 留意事項</b> .....	<b>23</b>

# 1 xStream API

## 1.1 概要

API (application programming interface) は、本サービスの xStream にプログラムからアクセスする場合に必要となります。

xStream API は、HTTP (Hypertext Transfer Protocol)、JSON (JavaScript Object Notation)、一般に利用可能な暗号化アルゴリズムなどの標準技術を使用しています。

xStream API は、RESTful に設計されています。すべての API リクエストは、HTTP の GET / POST / PUT / DELETE を使用して、URI (Uniform Resource Identifier) に対して実行されます。レスポンスは、JSON または XML (Extensible Markup Language) で返されます。

セキュリティは複数の方法で実現されています。認証には共有秘密鍵(キーペア)認証を使用します。経路のセキュリティ対策としては SSL (Secure Sockets Layer) を使用します。

## 1.2 本ガイドの対象

本ガイドは、xStream をプログラムに組み込みたい開発者/運用者向けです。読者は、API 利用やその標準技術に精通していることを前提としています。

## 1.3 改訂履歴

以下は、本ユーザーガイドの改訂履歴です。

- Document Revision: XAPI4.4.0-01 - Release Date: July 1, 2018
  - 初版リリース
- Document Revision: XAPI4.4.0-02 - Release Date: October 3, 2018
  - 一部表記の修正
- Document Revision: XAPI4.9.0-01 - Release Date: May 12, 2019
  - xStream4.9.0 に対応
- Document Revision: XAPI4.9.0-02 - Release Date: May 20, 2019
  - セキュリティプロトコルに関する注意事項を追記

- Document Revision: XAPI4.9.0-03 - Release Date: February 3, 2020
  - Python3 に対応
- Document Revision: XAPI4.15.0-01 - Release Date: September 7, 2020
  - xStream4.15.0 に対応

## 1.4 商標について

- Virtustream, Virtustream  $\mu$ VM、その他の Virtustream 製品の名称および製品名は、米国 Virtustream の、米国およびその他の国における商標または登録商標です。
- VMware および VMware の製品名は、VMware, Inc.の米国および各国での商標または登録商標です。
- その他記載されている製品名などの固有名詞は、各社の商標または登録商標です。
- 記載されている製品名等には、必ずしも商標表示((R),TM)を付記していません。

## 2 API リクエストの発行

### 2.1 エンドポイント

xStream API のエンドポイントは次の通りです。

```
https://<ベース URI>/api/<各エンドポイントのパス>
```

ベース URI はデータセンター毎に異なります。

データセンター	ベース URI
プライマリーデータセンター向け	console-v-jp1.ecl.ntt.com/
セカンダリーデータセンター向け	console-v-jp2.ecl.ntt.com/

ベース URI の後に続けて各エンドポイントのパスを記述します。

(例)

```
https://console-v-jp1.ecl.ntt.com/api/v1.3/HardwareTemplate
```

**Note:** セキュリティプロトコルは TLS1.2 以上をご使用ください。

### 2.2 API の認証方法

xStream API は、HTTP ヘッダーに含まれる Authorization ヘッダーを用いて認証します。

Authorization ヘッダーはキーペア用いて、API リクエスト毎に作成する必要があります。

Authorization ヘッダーを作成するためには、事前にキーペアの発行と有効化が完了している必要があります。

キーペアの発行と有効化は、xStream ポータルで実施します。

操作方法の詳細については、xStream Portal User Guide を参照してください。

#### 2.2.1 PYTHON を用いた AUTHORIZATION ヘッダーの生成例

以下は、Python を用いて、Authorization ヘッダーを生成するサンプルコードです。



```
headers = make_auth_headers(xstream_api_public_key, xstream_api_private_key,  
urlparse(xstream_request_uri).path, "")
```

### 2.2.2 POWERSHELL を用いた AUTHORIZATION ヘッダーの生成例

以下は、PowerShell を用いて、Authorization ヘッダーを生成するサンプルコードです。

当社では、下記環境にて API の動作確認をしています。

OS: Windows Server 2019 DataCenter Edition

Windows Power Shell: Power Shell 6.2

```
#Requires -Version 3  
Set-StrictMode -Version 2.0  
  
# TLSv1.2  
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12  
  
$ErrorActionPreference = "Stop"  
  
#Import [System.Web.HttpUtility]::UrlEncode  
Add-Type -AssemblyName System.Web  
  
function ToISO8601([DateTime] $date) {  
    return $date.ToString("yyyy-MM-ddTHH:mm:ssZ");  
}  
  
function Base64Encode([String] $s) {  
    if ([String]::IsNullOrEmpty($s)) {  
        return "";  
    };  
  
    return [Convert]::ToBase64String([Text.Encoding]::UTF8.GetBytes($s));  
}  
  
function Base64Decode([String] $s) {  
    return [Text.Encoding]::UTF8.GetString([Convert]::FromBase64String($s));  
}  
  
function HMACSHA256_Base64Encode([byte[]] $secret, [String] $message) {  
    $hmac = New-Object System.Security.Cryptography.HMACSHA256;  
    $hmac.Key = $secret;
```

```
[byte[]] $signature = $hmac.ComputeHash([Text.Encoding]::UTF8.GetBytes($message));

return [Convert]::ToBase64String($signature);
}

function KeyPairCredential([String] $key, [String] $secret) {
    return @{ AuthenticationScheme = "KeyPair";
        Key = $key;
        Secret = [Text.Encoding]::UTF8.GetBytes($secret);
    };
}

function SetKeyPairAuthHeader([Hashtable] $headers, [Hashtable] $keyPairCredential, [String]
$resource_path, [String] $body = $null){
    [String] $iso8601_timestamp = ToISO8601([DateTime]::UtcNow);
    [String] $clear =
[System.Web.HttpUtility]::UrlEncode("$( $keyPairCredential.Key)|$resource_path|$iso8601_timestam
p|$(Base64Encode $body)");
    [String] $signature = HMACSHA256_Base64Encode $keyPairCredential.Secret $clear;

    $headers.Add("Authorization", "KeyPair " + (Base64Encode $signature));
    $headers.Add("X-Public-Key", $keyPairCredential.Key);
    $headers.Add("X-Timestamp", $iso8601_timestamp);
    $headers.Add("URIPath", $resource_path);
}

function RestMethod([Hashtable] $xservice, [String] $pathAndQuery, [String] $body = $null) {
    [Uri] $uri = $xservice.BaseUrl + $pathAndQuery;
    [Hashtable] $credential = $xservice.Credential;
    [Hashtable] $headers = @{};

    SetKeyPairAuthHeader $headers $credential $uri.AbsolutePath $body;
}

$xstream_request_uri = 'https://console-v-jp1.ecl.ntt.com/api/v1.3';
$xstream_api_public_key = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx'; // 公開鍵を入力
$xstream_api_private_key = 'yyyyyyyy-yyy-yyy-yyy-yyy-yyy-yyy-yyy-yyy-yyy-yyy-yyy-yyy-
yyyyyyyy'; // 秘密鍵を入力
$endpoint_path = 'zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz'; // エンドポイントのパスを入力
```



```
[Hashtable] $xservice = @{
    BaseUrl = $xstream_request_uri;
    Credential = KeyPairCredential $xstream_api_public_key $xstream_api_private_key;
}

RestMethod $xservice $endpoint_path;
```

## 2.3 Python を用いた API リクエストの発行例

以下は、Python を用いて、HTTP ヘッダーに Authorization ヘッダーを埋め込み、API リクエストを発行する例です。

シナリオは仮想マシン(virtual-machine-id : ce69bfa5-cc62-2c84-fabb-65bb16724861)の Power ON と、その実行結果の確認です。仮想マシンの virtual-machine-id は、仮想マシン一覧の取得の API リクエストで取得できます。

当社では、下記環境にて API の動作確認をしています。

OS: Windows Server 2019 DataCenter Edition

Python: Python3.7.6

### 1. 仮想マシンの Power ON を行う「poweron.py」を準備します

```
# coding:utf-8

# Python example script for xStream auth using keypair
# Requires requests library to be present
import base64
import datetime
import hashlib
import hmac
import re
import urllib.request, urllib.parse, urllib.error
import json
from urllib.parse import urlparse
import requests

def make_auth_headers(key, secret, resource_path, body="", auth_method='KeyPair'):
    iso8601_timestamp = datetime.datetime.utcnow().isoformat() + "Z"
    clear = urllib.parse.quote_plus('|'.join([key, resource_path, iso8601_timestamp,
    base64.b64encode(body.encode('UTF-8')).decode() if body else ""]))
```

```

clear = re.sub(r'(%[A-F0-9]{2})', lambda x: x.group(0).lower(), clear)
signature = base64.b64encode(hmac.new(secret.encode('UTF-8'), clear.encode('UTF-8'),
digestmod=hashlib.sha256).digest())
return {'Authorization': 'KeyPair ' + base64.b64encode(signature).decode(), 'X-Public-Key': key,
        'X-Timestamp': iso8601_timestamp,
        'URIPath': resource_path}

def signed_common(uri, key, secret, reqfn, body="", verify_cert=True, auth_method='KeyPair'):
    bjson = ""
    extraarg = {}
    if body != "":
        bjson = json.dumps(body)
        extraarg['json'] = body
    headers = make_auth_headers(key, secret, urlparse(uri).path, bjson, auth_method)
    headers['Accept'] = 'application/json'
    res = reqfn(uri, headers=headers, verify=verify_cert, **extraarg)
    res.raise_for_status()
    return res.content

def signed_get(uri, key, secret, verify_cert=True, auth_method='KeyPair'):
    headers = make_auth_headers(key, secret, urlparse(uri).path, "", auth_method)
    headers['Accept'] = 'application/json'
    res = requests.get(uri, headers=headers, verify=verify_cert)
    return res.content

xstream_api_public_key = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx" // 公開鍵を入力
xstream_api_private_key = "yyyyyyyy-yyy-yyy-yyy-yyyyyyyyyyyyyyyyyyyy-yyy-yyy-yyy-
yyyyyyyyyyyy" // 秘密鍵を入力
xstream_request_uri = "https://console-v-jp1.ecl.ntt.com/api/v1.3/VirtualMachine/ce69bfa5-cc62-
2c84-fabb-65bb16724861/PowerOn" // URI を入力
verify_https_cert = True

# Set request type For POST PUT DELETE types. For GET use signed_get function
# requests.put requests.delete requests.post are valid
r_type = requests.post

# Correct payload must be passed in order to get successful response
req_body = "

```

```
# Post response
post_response = signed_common(xstream_request_uri, xstream_api_public_key,
xstream_api_private_key, r_type, req_body, verify_https_cert)
post_response = post_response.decode()
print (post_response)
```

## 2. 「poweron.py」を実行して API リクエストを発行し、レスポンスから MessageID を取得します

```
localhost$ python poweron.py | python -mjson.tool
{
  "Headers": {
    "MessageId": "bd18bab6-a768-49cb-8046-5fe87ec201b9"
  },
  "Status": "Queued"
}
```

## 3. MessageID を元に、API リクエストの実行結果を確認する「taskinfo.py」を準備します

```
# coding:utf-8

# Python example script for xStream auth using keypair
# Requires requests library to be present
import base64
import datetime
import hashlib
import hmac
import re
import urllib.request, urllib.parse, urllib.error
import json
from urllib.parse import urlparse
import requests

def make_auth_headers(key, secret, resource_path, body="", auth_method='KeyPair'):
    iso8601_timestamp = datetime.datetime.utcnow().isoformat() + "Z"
    clear = urllib.parse.quote_plus('|'.join([key, resource_path, iso8601_timestamp,
base64.b64encode(body.encode('UTF-8')).decode() if body else ""]))
    clear = re.sub(r'(%[A-F0-9]{2})', lambda x: x.group(0).lower(), clear)
    signature = base64.b64encode(hmac.new(secret.encode('UTF-8'), clear.encode('UTF-8'),
digestmod=hashlib.sha256).digest())
    return {'Authorization': 'KeyPair ' + base64.b64encode(signature).decode(), 'X-Public-Key': key,
```

```

        'X-Timestamp': iso8601_timestamp,
        'URIPath': resource_path}

def signed_common(uri, key, secret, reqfn, body="", verify_cert=True, auth_method='KeyPair'):
    bjson = ""
    extraarg = {}
    if body != "":
        bjson = json.dumps(body)
        extraarg['json'] = body
    headers = make_auth_headers(key, secret, urlparse(uri).path, bjson, auth_method)
    headers['Accept'] = 'application/json'
    res = reqfn(uri, headers=headers, verify=verify_cert, **extraarg)
    res.raise_for_status()
    return res.content

def signed_get(uri, key, secret, verify_cert=True, auth_method='KeyPair'):
    headers = make_auth_headers(key, secret, urlparse(uri).path, "", auth_method)
    headers['Accept'] = 'application/json'
    res = requests.get(uri, headers=headers, verify=verify_cert)
    return res.content

xstream_api_public_key = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx" // 公開鍵を入力
xstream_api_private_key = "yyyyyyyy-yyy-yyy-yyy-yyyyyyyyyyyyyyyyyyyy-yyy-yyy-yyy-
yyyyyyyyyyyy" // 秘密鍵を入力
xstream_request_uri = "https://console-v-jp1.ecl.ntt.com/api/v1.3/TaskInfo/bd18bab6-a768-49cb-
8046-5fe87ec201b9" // URI に MessageID を指定
verify_https_cert = True

# Get response
get_response = signed_get(xstream_request_uri, xstream_api_public_key, xstream_api_private_key,
verify_https_cert)
post_response = post_response.decode()
print (get_response)

```

#### 4. 「taskinfo.py」を実行し、レスポンスから"Success"を確認します

```

localhost $ python taskinfo.py | python -mjson.tool
{
  "Acl": [
    {

```

```
"Permissions": 1,
"ResourceRole": "3f9d5bf4-0f63-4ede-8a31-xxxxxxxxxxxx",
"UserGroup": "9c942b2d-ef5f-42d6-950b-xxxxxxxxxxxx"
},
{
  "Permissions": 1,
  "ResourceRole": "3985741b-0df2-4e9a-869b-xxxxxxxxxxxx",
  "UserGroup": "42ab4387-7241-44c2-9d1c-xxxxxxxxxxxx"
},
{
  "Permissions": 1,
  "ResourceRole": "3985741b-0df2-4e9a-869b-xxxxxxxxxxxx",
  "UserGroup": "9c942b2d-ef5f-42d6-950b-xxxxxxxxxxxx"
}
],
"Cancellable": false,
"CorrelationId": "bd18bab6-a768-49cb-8046-xxxxxxxxxxxx",
"CreatedBy": "example_user@com.local",
"CreatedDate": "2018-05-23T09:46:04.202Z",
"CustomerId": 0,
"EditedBy": null,
"EditedDate": "2018-05-23T09:46:08.041Z",
"Errors": null,
"FinishTime": "2018-05-23T09:46:06.07Z",
"HypervisorID": "2b756764-1c6d-43f1-87a1-xxxxxxxxxxxx",
"LocalizableError": null,
"ManagedResourceID": "ce69bfa5-cc62-2c84-fabb-xxxxxxxxxxxx",
"ManagedResourceIdentifier": "vm-xxxx",
"ManagedResourceName": "DS-IZ-SuSE1",
"ManagedResourceType": "VirtualMachine",
"MessageName": "PowerOn",
"ParentTaskId": null,
"PortalUserName": "example_user@com.local",
"Progress": 100,
"Result": null,
"SessionID": null,
"SiteID": null,
"StartTime": "2018-05-23T09:46:05.304Z",
"State": 4,
"StateMessage": "Success", // タスク実行が成功した場合"Success"、失敗した場合"Error"等が表示される
"TaskId": "bd18bab6-a768-49cb-8046-5fe87ec201b9",
```

```
"TaskInfoID": "bd18bab6-a768-49cb-8046-5fe87ec201b9",  
"TaskName": "Power On virtual machine",  
"TenantID": "41fe8893-db97-4e6b-bde8-xxxxxxxxxxxx",  
"UserName": ""  
}
```

## 3 本サービスで利用可能なAPI一覧

本サービスで利用可能なAPIを以下に記載します。

### 3.1 仮想マシン一覧の取得

本操作は、お客様テナント内の仮想マシン一覧を取得します。

リクエスト:

```
GET /api/v1.3/VirtualMachine
```

レスポンス:

```
[
  { //1 台目の VM の情報
    -----中略-----
    "Name": "AAAAJP014XVM999", // VM の RID
    -----中略-----
    "PowerState": "poweredOff", // VM の電源状態
    -----中略-----
    "VirtualMachineID": "06640105-a4e2-8add-9ce4-xxxxxxxxxxxx" // 仮想マシンの ID (Virtual Machine ID)
  },
  { //2 台目の VM の情報、以下 VM 情報分繰り返す
    -----中略-----
  }
]
```

### 3.2 仮想マシン情報の取得

本操作は、対象の仮想マシンの情報を取得します。

リクエスト:

```
GET /api/v1.3/VirtualMachine/{virtual-machine-id}
```

レスポンス:

```
{ //該当 VM の情報
  -----中略-----
  "Name": "AAAAJP014XVM999", // VM の RID
  -----中略-----
}
```

```
"PowerState": "poweredOff", // VM の電源状態
-----中略-----
"VirtualMachineID": "06640105-a4e2-8add-9ce4-xxxxxxxxxxxx" // 仮想マシンの ID(Virtual Machine ID)
}
```

### 3.3 仮想マシンのパフォーマンス情報の取得

本操作は、Virtual Machine IDs で指定した仮想マシンのパフォーマンス情報を取得します。  
CPU/メモリ/ストレージ/NW の 1 時間平均の消費量が確認できます。

リクエスト:

POST /api/v1.3/VmComputeConsumption

HTTP Request Body

```
{
  "DateFrom": "2018-03-01T00:00:00Z",
  "DateTo": "2018-03-06T00:00:00Z",
  "VirtualMachineIDs": ["501059eb-9a31-191a-93e0-cbf03d9c9cc9", "cbd9b97b-b6a7-7535-18ab-94971c0c2911"]
}
```

レスポンス:

```
{
  "Headers": {
    "Date": "2018-01-10T00:00:00Z",
    "ProfileID": "11ba71e4-278c-4d8e-8841-8aab2952d93d",
    "TotalUvmh": 0.85996,
    "TotalUvmhSeconds": 86400,
    "CpuUvmh": 0.85996,
    "NetUvmh": 0.03804,
    "MemUvmh": 0.38691,
    "IoUvmh": 0.0746,
    "AvgCpuMhz": 440.376648,
    "AvgMemKb": 232467.9,
    "AvgNetKbps": 0.9745763,
    "AvgReadIops": 46,
```



```
"AvgWriteIops": 5,  
  "AvgIops": 51  
},  
  
"Status": "Queued"  
  
}
```

(参考) μVM 4 要素の確認箇所:

AvgCpuMhz : CPU 消費量(MHz 単位)

AvgMemKb : メモリ消費量(KB 単位)

AvgIops : 仮想マシンに接続された全てのディスクの合計ストレージ消費量(IOPS 単位)

AvgNetKbps : 仮想マシンに接続された全てのネットワークインターフェースの合計スループット消費量(KB/sec 単位)

### 3.4 仮想マシンの Power ON

本操作は、仮想マシンを Power ON します。

リクエスト:

```
POST /api/v1.3/VirtualMachine/{virtual-machine-id}/PowerOn
```

レスポンス:

```
{  
  
  "Headers": {  
  
    "MessageId": "88cc3415-9374-4892-90d0-135a8a0345fc"  
  },  
  
  "Status": "Queued"  
  
}
```

### 3.5 仮想マシンの Power OFF

本操作は、仮想マシンを Power OFF します。

リクエスト:

```
POST /api/v1.3/VirtualMachine/{virtual-machine-id}/PowerOff
```

レスポンス:

```
{  
  
  "Headers": {  
  
    "MessageId": "88cc3415-9374-4892-90d0-135a8a0345fc"  
  
  },  
  
  "Status": "Queued"  
  
}
```

### 3.6 OS シャットダウン

本操作は、仮想マシン上の OS をシャットダウンします。

ハイパーバイザー管理ツールがインストールされた仮想マシンのみサポートされます。

リクエスト:

```
POST /api/v1.3/VirtualMachine/{virtual-machine-id}/ShutdownOS
```

レスポンス:

```
{  
  
  "Headers": {  
  
    "MessageId": "88cc3415-9374-4892-90d0-135a8a0345fc"  
  
  },  
  
  "Status": "Queued"  
  
}
```

### 3.7 ハードウェアテンプレート一覧の取得

本操作は、本サービスで提供しているハードウェアテンプレート一覧を取得します。

リクエスト:

```
GET /api/v1.3/HardwareTemplate
```

レスポンス:

```
[
  -----中略-----
  {
    "Description": "4vCPU 8Memory",
    "HardwareTemplateID": "b348b3e2-8ce0-4da5-b45f-7b3dfcac1531", // ハードウェアテンプレートの ID
    "Name": "medium2", // ハードウェアテンプレート名
    "NumCpu": 4, // vCPU 数
    "RamAllocatedMB": 8192, // メモリサイズ[MB]
    "TenantID": "b8980de8-c4c8-47bf-bdbb-xxxxxxxxxxxx"
  },
  -----中略-----
]
```

### 3.8 ハードウェアテンプレートの変更

本操作は、既存の仮想マシンのハードウェアテンプレート(vCPU 数、メモリサイズ)を変更します。

HardwareTemplateID パラメーターで、ハードウェアテンプレートを指定します。

リクエスト:

```
POST /api/v1.3/VirtualMachine/ReconfigureHardware
```

HTTP Request Body

```
{
  VirtualMachineID : string, // 仮想マシンの ID
  HardwareTemplateID : string // ハードウェアテンプレートの ID
}
```

レスポンス:

```
{
  "Headers": {
```

```
"MessageId": "88cc3415-9374-4892-90d0-135a8a0345fc"

},

"Status": "Queued"

}
```

### 3.9 キーペア失効日時の設定

本操作は、xStream ユーザーアカウントのキーペアの失効日時を設定します。

リクエスト:

POST /api/v1.3/User/SetKeyPairExpiry

HTTP Request Body

```
{
  ExpirationDateUTC : DateTime, // キーペアの失効日時をU T C形式で指定
  PublicKey : string // 対象キーペアの公開鍵を指定
}
```

レスポンス:

```
{

"Headers": {

"MessageId": "32a7dfe5-4b8d-471b-b329-948337809904"

},

"Status": "Queued"

}
```

### 3.10 タスク実行結果の取得

本操作は、API リクエストのレスポンスで返される MessageID を用いて、該当 API リクエストの実行結果を取得します。

リクエスト:

```
GET /api/v1.3/TaskInfo/{MessageId}
```

レスポンス:

```
{
  -----中略-----
  "Cancellable": false,
  "CorrelationId": "bd18bab6-a768-49cb-8046-xxxxxxxxxxxx",
  "CreatedBy": "example_user@com.local",
  "CreatedDate": "2018-05-23T09:46:04.202Z",
  "CustomerId": 0,
  "EditedBy": null,
  "EditedDate": "2018-05-23T09:46:08.041Z",
  "Errors": null,
  "FinishTime": "2018-05-23T09:46:06.07Z",
  "HypervisorID": "2b756764-1c6d-43f1-87a1-xxxxxxxxxxxx",
  "LocalizableError": null,
  "ManagedResourceID": "ce69bfa5-cc62-2c84-fabb-xxxxxxxxxxxx",
  "ManagedResourceIdentifier": "vm-xxxx",
  "ManagedResourceName": "DS-IZ-SuSE1",
  "ManagedResourceType": "VirtualMachine",
  "MessageName": "PowerOn",
  "ParentTaskId": null,
  "PortalUserName": "example_user@com.local",
  "Progress": 100,
  "Result": null,
  "SessionID": null,
  "SiteID": null,
  "StartTime": "2018-05-23T09:46:05.304Z",
  "State": 4,
  "StateMessage": "Success", // タスク実行が成功した場合"Success"、失敗した場合"Error"等が表示される
  "TaskId": "bd18bab6-a768-49cb-8046-5fe87ec201b9",
  "TaskInfoID": "bd18bab6-a768-49cb-8046-5fe87ec201b9",
  "TaskName": "Power On virtual machine",
  "TenantID": "41fe8893-db97-4e6b-bde8-xxxxxxxxxxxx",
  "UserName": ""
}
```

}

## 4 留意事項

- プラットフォームについて  
API リクエストを発行するお客様環境(インターネット接続環境、OS、言語、ツールやその組み合わせなど)に関して、制限は設けておりません。  
ただし、API リクエストが、お客様環境に依存して正しく発行されない場合は、ご利用の環境の変更をお願いする場合があります。  
また、お客様環境関連の問い合わせについては、当社は回答できない場合があります。
- サポートする API について  
本マニュアルに記載のない API コマンドは使用しないでください。  
本マニュアルに記載のない API コマンドを使用したことによるお客様テナント内に生じた不具合等について、当社は一切のサポートを提供いたしません。  
また、当社サービスに不具合等の影響が生じた場合は、その損害額を請求する場合があります。
- 仕様の変更について  
API の仕様は予告なく変更する場合があります。