aws

開發人員指南

NICE DCV DCV 會話管理器

NICE DCV DCV 會話管理器: 開發人員指南

Table of Contents

什麼是 NICE DCV 會話管理器？

NICE DCV 工作階段管理器是一組可安裝的軟體套件 (代理程式和代理程式) 和應用程式設計介面 (API)，可讓開發人員和獨立軟體廠商 (ISV) 輕鬆建立前端應用程式，以程式設計方式在 NICE DCV 伺服器叢集中建立和管理 NICE DCV 工作階段的生命週期。

本指南說明如何使用工作階段管理員 API 來管理 NICE DCV 工作階段的生命週期。如需有關如何安裝和設定工作階段管理員代理程式和代理程式的詳細資訊，請參閱 NICE DCV 工作階段管理員管理員指南。

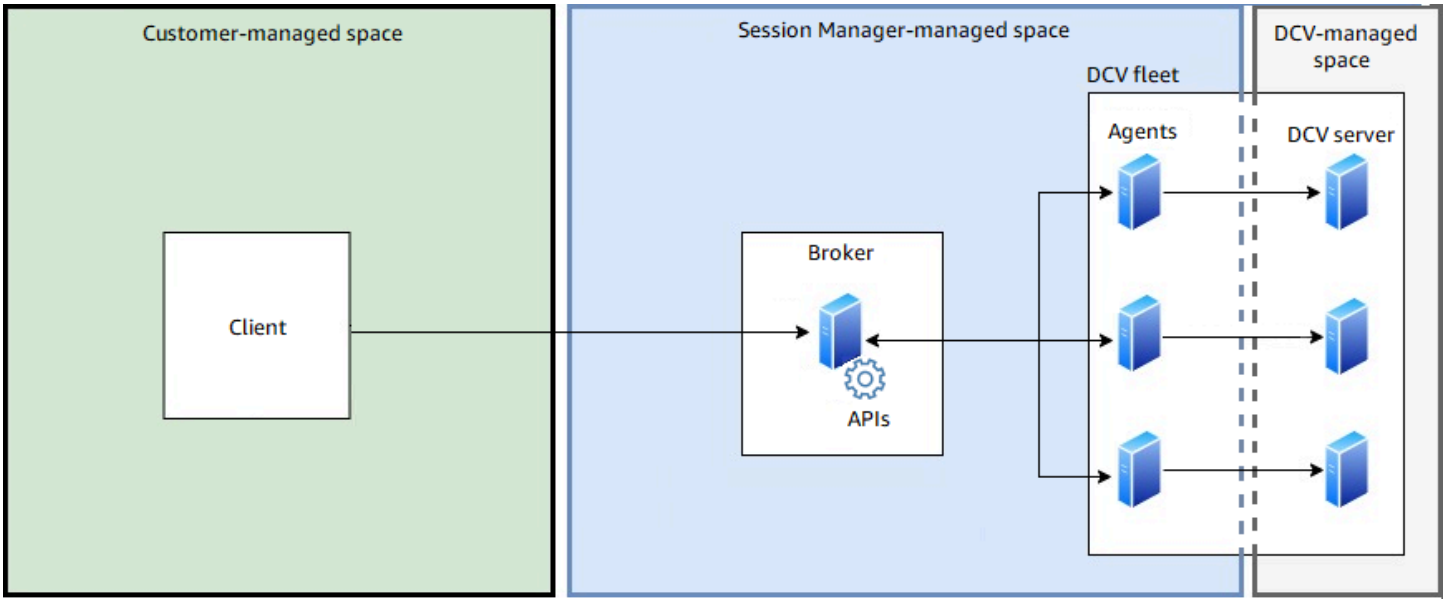
先決條件

開始使用工作階段管理員 API 之前，請確定您已熟悉 NICE DCV 和 NICE DCV 工作階段。如需詳細資訊，請參閱 NICE DCV 管理員指南。

主題

Session Manager 的運作方式

下列的圖表顯示了 Session Manager 的高級元件。



中介裝置

代理程式是託管和公開工作階段管理員 API 的 Web 伺服器。它會從用戶端接收並處理 API 要求以管理 NICE DCV 工作階段，然後將指示傳送給相關代理程式。代理程式必須安裝在與 NICE DCV 伺服器分開的主機上，但用戶端必須可存取代理程式，而且必須能夠存取代理程式。

代理程式

代理程式會安裝在叢集中的每個 NICE DCV 伺服器上。代理程式會從代理程式接收指示，並在各自的 NICE DCV 伺服器上執行這些指示。代理程式也會監控 NICE DCV 伺服器的狀態，並將定期狀態更新傳送回代理程式。

API

會話管理器公開了一組 REST 應用程序編程接口（API），可用於管理 NICE DCV 服務器隊列上的 NICE DCV 會話。這些 API 託管在代理上並由公開。開發人員可以建置呼叫 API 的自訂工作階段管理用戶端。

客戶端

用戶端是您開發用來呼叫 Broker 公開的工作階段管理員 API 的前端應用程式或入口網站。使用者可以使用用戶端來管理叢集中 NICE DCV 伺服器上託管的工作階段。

訪問令牌

為了發出 API 請求，您必須提供訪問令牌。可以透過註冊的用戶端 API 從代理程式或外部授權伺服器要求權杖。要請求和訪問令牌，客戶端 API 必須提供有效的憑據。

用戶端 API

用戶端 API 是從工作階段管理員 API 定義 YAML 檔案產生的，使用 Swagger Codegen。客戶端 API 用於發出 API 請求。

NICE DCV DCV 會議

您必須在用戶端可以連線的 NICE DCV 伺服器上建立 NICE DCV 工作階段。用戶端只能在有作用中的工作階段時連線到 NICE DCV 伺服器。NICE DCV 支援主控台和虛擬工作階段。您可以使用工作階段管理員 API 來管理 NICE DCV 工作階段的生命週期。NICE DCV 工作階段可為下列任一種狀態：

- CREATING— 代理人正在建立工作階段。
- READY工作階段已準備好接受用戶端連線。
- DELETING正在刪除工作階段。
- DELETED已刪除工作階段。

- UNKNOWN無法判斷工作階段的狀態。代理程式和代理程式可能無法通訊。

功能

DCV Session Manager 提供如下功能：

- 提供 NICE DCV 工作階段資訊 — 取得在多個 NICE DCV 伺服器上執行之工作階段的相關資訊。
- 管理多個 NICE DCV 工作階段的生命週期 — 透過一個 API 請求，跨多個 NICE DCV 伺服器建立或刪除多個使用者的多個工作階段。
- 支援標籤 — 建立工作階段時，使用自訂標籤以 NICE DCV 伺服器群組為目標。
- 管理多個 NICE DCV 工作階段的權限 — 使用一個 API 請求修改多個工作階段的使用者權限。
- 提供連線資訊 — 擷取 NICE DCV 工作階段的用戶端連線資訊。
- 支援雲端和內部部署 — 在內部部署AWS、內部部署或搭配替代雲端伺服器使用工作階段管理員。

入門

本節顯示如何開始使用工作階段管理員 API。

在本節中，我們將顯示如何使用來做到這一點，使用DescribeSessions API 做到這一點。

主題

- 產生用戶端 API
- 登錄您的 API 用戶端
- 獲取訪問令牌並發出 API 請求

產生用戶端 API

工作階段管理員 API 是在單一 YAML 檔案中定義的。這些 API 是基於 OpenAPI3.0 規範，它定義了一個與 RESTful API 無關語言的標準接口。如需詳細資訊，請參閱 OpenAPI 規格。

使用 YAML 檔案，您可以使用其中一種支援的語言產生 API 用戶端。若要這樣做，您必須使用施瓦格編碼 3.0 或更新版本。如需支援的語言的詳細資訊，請參閱 Sagger-codegen 存放庫。

若要產生 API 用戶端

1. 從工作階段管理員代理程式下載工作階段管理員 API YAML 檔案。YAML 檔案可從下列網址取得。

   ```
   https://broker_host_ip:port/dcv-session-manager-api.yaml
   ```

2. 安裝施瓦格代碼.

   - macOS

     ```
     $ brew install swagger-codegen
     ```

   - 其他平台

     ```
     $ git clone https://github.com/swagger-api/swagger-codegen --branch 3.0.0
     ```

     ```
     $ cd swagger-codegen
     ```

3. 產生 API 用戶端。

   - macOS

     ```
     $ swagger-codegen generate -i /path_to/yaml_file -l language -o $output_folder
     ```

   - 其他平台

     ```
     $ mvn clean package
     ```

     ```
     $ java -jar modules/swagger-codegen-cli/target/swagger-codegen-cli.jar generate -
     i /path_to/yaml_file -l language -o output_folder
     ```

登錄您的 API 用戶端

要發出 API 請求，您必須首先從代理商中檢索訪問令牌。要從代理商獲取訪問令牌，您必須向代理提供客戶端 API 的憑據。憑據基於客戶端向代理註冊時生成的客戶 ID 和客戶端密碼。如果您沒有用戶端 ID 和用戶端密碼，則必須向 Broker 管理員提出要求。如需向 Broker 註冊用戶端 API 以及取得用戶端 ID 和密碼的詳細資訊，請參閱register-api-client。

獲取訪問令牌並發出 API 請求

首先，我們導入應用程序所需的模型。

然後我們宣告用戶端 ID (__CLIENT_ID)、用戶端密碼 (__CLIENT_SECRET) 和代理人 URL 的變數，包括連接埠號碼 (__PROTOCOL_HOST_PORT)。

接下來，我們創建一個名為生build_client_credentials成客戶端憑據的函數。若要產生用戶端認證，您必須先串連用戶端 ID 和用戶端密碼，並使用冒號 (*client_ID*:*client_password*) 分隔值，然後 Base64 對整個字串進行編碼。

```
import swagger_client
import base64
import requests
import json
from swagger_client.models.describe_sessions_request_data import
 DescribeSessionsRequestData
from swagger_client.models.key_value_pair import KeyValuePair
from swagger_client.models.delete_session_request_data import DeleteSessionRequestData
```

```
from swagger_client.models.update_session_permissions_request_data import
 UpdateSessionPermissionsRequestData
from swagger_client.models.create_session_request_data import CreateSessionRequestData

__CLIENT_ID = '794b2dbb-bd82-4707-a2f7-f3d9899cb386'
__CLIENT_SECRET = 'MzcxNzJhN2UtYjEzNS00MjNjLTg2N2YtMjFlZmRlZWNjMDU1'
__PROTOCOL_HOST_PORT = 'https://<broker-hostname>:8443'

def build_client_credentials():
    client_credentials = '{client_id}:{client_secret}'.format(client_id=__CLIENT_ID,
 client_secret=__CLIENT_SECRET)
    return base64.b64encode(client_credentials.encode('utf-8')).decode('utf-8')
```

現在我們有了客戶端憑據，我們可以使用它從代理請求訪問令牌。要做到這一點，我們創建一個名為的函數get_access_token。您必須呼叫POST on https://*Broker_IP*:*8443*/oauth2/token?grant_type=client_credentials，並提供授權標頭，其中包括 BASIC 編碼的用戶端認證，以及的內容類型application/x-www-form-urlencoded。

```
def get_access_token():
    client_credentials = build_client_credentials()
    headers = {
        'Authorization': 'Basic {}'.format(client_credentials),
        'Content-Type': 'application/x-www-form-urlencoded'
    }
    endpoint = __PROTOCOL_HOST_PORT + '/oauth2/token?grant_type=client_credentials'
    print('Calling', endpoint, 'using headers', headers)
    res = requests.post(endpoint, headers=headers, verify=True)
    if res.status_code != 200:
        print('Cannot get access token:', res.text)
        return None
    access_token = json.loads(res.text)['access_token']
    print('Access token is', access_token)
    return access_token
```

現在，我們創建實例化客戶端 API 所需的函數。要實例化客戶端 API，您必須指定客戶端配置和要用於請求的標頭。此get_client_configuration函數會建立一個組態物件，其中包括 Broker 的 IP 位址和連接埠，以及 Broker 自我簽署憑證的路徑 (您應該已經從 Broker 管理員那裡接收)。該set_request_headers函數創建一個請求頭對象，其中包括客戶端憑據和訪問令牌。

```
def get_client_configuration():
```

```
    configuration = swagger_client.Configuration()
    configuration.host = __PROTOCOL_HOST_PORT
    configuration.verify_ssl = True
    # configuration.ssl_ca_cert = cert_file.pem
    return configuration

def set_request_headers(api_client):
    access_token = get_access_token()
    api_client.set_default_header(header_name='Authorization',
                                  header_value='Bearer {}'.format(access_token))

def get_sessions_api():
    api_instance =
 swagger_client.SessionsApi(swagger_client.ApiClient(get_client_configuration()))
    set_request_headers(api_instance.api_client)
    return api_instance
```

最後，我們建立呼叫DescribeSessions API 的 main 方法。如需詳細資訊，請參閱 DescribeSessions。

```
def describe_sessions(session_ids=None, next_token=None, tags=None, owner=None):
    filters = list()
    if tags:
        for tag in tags:
            filter_key_value_pair = KeyValuePair(key='tag:' + tag['Key'],
 value=tag['Value'])
            filters.append(filter_key_value_pair)
    if owner:
        filter_key_value_pair = KeyValuePair(key='owner', value=owner)
        filters.append(filter_key_value_pair)

    request = DescribeSessionsRequestData(session_ids=session_ids, filters=filters,
 next_token=next_token)
    print('Describe Sessions Request:', request)
    api_instance = get_sessions_api()
    api_response = api_instance.describe_sessions(body=request)
    print('Describe Sessions Response', api_response)

def main():
    describe_sessions(
        session_ids=['SessionId1895', 'SessionId1897'],
```

```
        owner='an owner 1890',
        tags=[{'Key': 'ram', 'Value': '4gb'}])
```

Session Manager API 參考

本節提供每個工作階段段段段段段 API 動作段落的說明、語法和使用範例。

主題

CloseServers

關閉一或多個 NICE DCV 伺服器。當您關閉 NICE DCV 伺服器時，會讓它無法用於 NICE DCV 工作階段放置。您無法在封閉的伺服器上建立 NICE DCV 工作階段。關閉伺服器可確保伺服器上沒有工作階段正在執行，且使用者無法在其上建立新的工作階段。

主題

請求參數

`ServerId`

要關閉的伺服器 ID。

類型：字串

必要：是

Force

強制關閉操作。如果您指定`true`，即使伺服器具有執行階段作業，也會關閉該伺服器。工作階段繼續運作。

類型：布林值

必要：否

回應參數

RequestId

要求的唯一 ID。

SuccessfulList

已成功關閉的 NICE DCV 伺服器的相關資訊。此資料結構包含下列巢狀回應參數：

ServerId

已順利關閉的伺服器識別碼。

UnsuccessfulList

無法關閉的 NICE DCV 伺服器的相關資訊。此資料結構包含下列巢狀回應參數：

CloseServerRequestData

原始要求失敗的相關資訊。此資料結構包含下列巢狀回應參數：

ServerId

無法關閉的 NICE DCV 伺服器的識別碼。

Force

請求的力參數。

FailureCode

失敗的代碼。

FailureReason

失敗的原因。

範例

Python

請求

下列範例會關閉兩個 NICE DCV 伺服器 (`serverId1`和`serverId2`)。伺服器`serverId2`不存在，導致失敗。

```
from swagger_client.models import CloseServerRequestData

def get_servers_api():
    api_instance =
 swagger_client.ServersApi(swagger_client.ApiClient(get_client_configuration()))
    set_request_headers(api_instance.api_client)
    return api_instance

def close_servers(server_ids):
    request = [CloseServerRequestData(server_id=server_id) for server_id in
 server_ids]
    print('Close Servers Request:', request)
    api_instance = get_servers_api()
    api_response = api_instance.close_servers(body=request)
    print('Close Servers Response:', api_response)
    open_servers(server_ids)

def main():
    close_servers(["serverId1", "serverId2"])
```

回應

以下為其輸出。

```
{
    "RequestId": "4d7839b2-a03c-4b34-a40d-06c8b21099e6",
    "SuccessfulList": [
        {
            "ServerId": "serverId1"
        }
    ],
    "UnsuccessfulList": [
        {
```

```
            "OpenServerRequestData": {
                "ServerId": "serverId2"
            },
            "FailureCode": "DCV_SERVER_NOT_FOUND",
            "FailureReason": "Dcv server not found."
        }
    ]
}
```

CreateSessions

使用指定的詳細資訊建立新 NICE DCV 工作階段段段。

API 動作

- 請求參數
- 回應參數
- 範例

請求參數

`Name`

工作階段的名稱。

類型：字串

必要：是

`Owner`

工作階段擁有者的名稱。必須是目標 NICE DCV 伺服器上現有使用者的名稱。

類型：字串

必要：是

`Type`

工作階段類型。如需有關工作階段類型的詳細資訊，請參閱《NICE DCV 管理員指南》中的 NICE DCV 工作階段簡介。

有效值：主控台 | 虛擬

類型：字串

必要：是

`InitFile`

支援在 Linux NICE DCV 伺服器上的虛擬工作階段。視窗和 Linux NICE DCV 伺服器上的主控台工作階段不支援此功能。NICE DCV 伺服器上用於初始化工作階段建立時執行的自訂指令碼路徑。檔案路徑與`agent.init_folder`代理程式組態參數指定的 init 目錄相對。如果檔案位於指定的 init 目錄中，請僅指定檔案名稱。如果檔案不在指定的 init 目錄中，請指定相對路徑。如需詳細資訊，請參閱 NICE DCV 工作階段管理員管理員指南中的代理程式設定檔。

類型：字串

必要：否

`MaxConcurrents`

NICE DCV 並行用戶端數上限。

類型：整數

必要：否

`DcvGlEnabled`

指出虛擬工作階段是否設定為使用以硬體為基礎的 OpenGL。僅支援虛擬工作階段。Windows NICE DCV 伺服器不支援此參數。

有效值：true | false

類型：布林值

必要：否

`PermissionsFile`

權限檔案的 base64 編碼內容。如果省略，則預設為伺服器預設值。如需詳細資訊，請參閱《NICE DCV 管理員指南》中的 < 設定 NICE DCV 授權 > 。

類型：字串

必要：否

EnqueueRequest

指出如果無法立即滿足請求，是否要將請求排入佇列。

類型：布林值

預設：false

必要：否

AutorunFile

支援 Windows NICE DCV 伺服器上的主控台工作階段，以及 Linux NICE DCV 伺服器上的虛擬工作階段。Linux NICE DCV 伺服器上的主控台工作階段不支援此功能。

主機伺服器上要在工作階段內執行之檔案的路徑。檔案路徑相對於為「agent.autorun_folder代理程式」組態參數指定的自動執行目錄。如果檔案位於指定的自動執行目錄中，請僅指定檔案名稱。如果檔案不在指定的自動執行目錄中，請指定相對路徑。如需詳細資訊，請參閱 NICE DCV 工作階段管理員管理員指南中的代理程式設定檔。

該文件代表指定的所有者運行。指定的擁有者必須具有在伺服器上執行檔案的權限。在 Windows NICE DCV 伺服器上，檔案會在擁有者登入工作階段時執行。在 Linux NICE DCV 伺服器上，檔案會在建立工作階段時執行。

類型：字串

必要：否

AutorunFileArguments

支援在 Linux NICE DCV 伺服器上的虛擬工作階段。在視窗和 Linux NICE DCV 伺服器上的主控台工作階段中不支援此功能。命令行參數AutorunFile在會話內執行時傳遞給。參數按照它們出現到給定數組的順序傳遞。可以配置允許的最大參數數量和允許的每個參數的最大長度。如需詳細資訊，請參閱《NICE DCV 工作階段管理員管理員指南》中的代理人組態檔。

類型：字串陣列

必要：否

DisableRetryOnFailure

指出建立工作階段要求在 NICE DCV 主機上因任何原因失敗後，是否不重試該要求。如需有關建立工作階段重試機制的詳細資訊，請參閱《NICE DCV 工作階段管理員指南》中的 Broker 組態檔案。

類型：布林值

預設：false

必要：否

Requirements

伺服器必須滿足才能放置工作階段的需求。這些需求可以包括伺服器標籤和/或伺服器屬性，伺服器標籤和伺服器屬性都是透過呼叫 DescribeServersAPI 來擷取的。

需求條件表達式：

- *##* ！ = *b* 如果 *a* 不等於 *b*，則為真
- 如果 *a* 等於 *b*，則*##* = *b* 真
- *##* > *b* 如果*##*大於 *b*，則為真
- 如果 *a* 大於或等於 *b*，則*##* > = *b* 真
- *#* < *b* 如果 *a* 小於 *b*，則為真
- 如果 *a* 小於或等於 *b*，則*##* <= *b* 真
- ***## a ##### b####***

要求布爾運算符：

- *a* 和 *b* 如果 *a* 和 *b* 是真的，則為真
- 如果 *a* 或 *b* 是真的，則 *a* 或 *b* 為真
- 如果 *a* 是假的*#*則不是真的

標籤鍵必須為前綴`tag:`，服務器屬性必須以前綴`server:`。要求表達式支持括號`()`。

需求示例：

- `tag:color = 'pink' and (server:Host.Os.Family = 'windows' or tag:color := 'red')`
- `"server:Host.Aws.Ec2InstanceType := 't2' and server:Host.CpuInfo.NumberOfCpus >= 2"`

可以使用指數表示法來指定數值，例如：`"server:Host.Memory.TotalBytes > 1024E6"`。

支援的伺服器屬性如下：

- `Id`
- `Hostname`

- Version
- SessionManagerAgentVersion
- Host.Os.BuildNumber
- Host.Os.Family
- Host.Os.KernelVersion
- Host.Os.Name
- Host.Os.Version
- Host.Memory.TotalBytes
- Host.Memory.UsedBytes
- Host.Swap.TotalBytes
- Host.Swap.UsedBytes
- Host.CpuLoadAverage.OneMinute
- Host.CpuLoadAverage.FiveMinutes
- Host.CpuLoadAverage.FifteenMinutes
- Host.Aws.Ec2InstanceId
- Host.Aws.Ec2InstanceType
- Host.Aws.Region
- Host.Aws.Ec2ImageId
- Host.CpuInfo.Architecture
- Host.CpuInfo.ModelName
- Host.CpuInfo.NumberOfCpus
- Host.CpuInfo.PhysicalCoresPerCpu
- Host.CpuInfo.Vendor

類型：字串

必要：否

StorageRoot

指定要用於儲存工作階段之資料夾的路徑。如需有關 NICE DCV 工作階段儲存的詳細資訊，請參閱《NICE DCV 管理員指南》中的啟用工作階段儲存。

類型：字串

必要：否

回應參數

Id

工作階段的唯一 ID。

Name

工作階段名稱。

Owner

工作階段擁有者。

Type

工作階段的類型。

State

工作階段的狀態。如果要求成功完成，工作階段就會進入CREATING狀態。

Substate

工作階段的子狀態。如果請求成功完成，子狀態將進入SESSION_PLACING子狀態。

範例

Python

請求

下列範例會建立三個工作階段。

```
from swagger_client.models.create_session_request_data import
 CreateSessionRequestData

def get_sessions_api():
    api_instance =
 swagger_client.SessionsApi(swagger_client.ApiClient(get_client_configuration()))
    set_request_headers(api_instance.api_client)
```

```
    return api_instance

def create_sessions(sessions_to_create):
    create_sessions_request = list()
    for name, owner, session_type, init_file_path, autorun_file,
 autorun_file_arguments, max_concurrent_clients,\
            dcv_gl_enabled, permissions_file, requirements, storage_root in
 sessions_to_create:
        a_request = CreateSessionRequestData(
            name=name, owner=owner, type=session_type,
            init_file_path=init_file_path, autorun_file=autorun_file,
 autorun_file_arguments=autorun_file_arguments,
 max_concurrent_clients=max_concurrent_clients,
            dcv_gl_enabled=dcv_gl_enabled, permissions_file=permissions_file,
 requirements=requirements, storage_root=storage_root)
        create_sessions_request.append(a_request)

    api_instance = get_sessions_api()
    print('Create Sessions Request:', create_sessions_request)
    api_response = api_instance.create_sessions(body=create_sessions_request)
    print('Create Sessions Response:', api_response)

def main():
    create_sessions([
    ('session1', 'user1', 'CONSOLE', None, None, None, 1, None, '/dcv/
permissions.file', "tag:os = 'windows' and server:Host.Memory.TotalBytes > 1024", "/
storage/root"),
    ('session2', 'user1', 'VIRTUAL', None, 'myapp.sh', None, 1, False, None, "tag:os
 = 'linux'", None),
    ('session3', 'user1', 'VIRTUAL', '/dcv/script.sh', 'myapp.sh', ['argument1',
 'argument2'], 1, False, None, "tag:os = 'linux'", None),
])
```

回應

以下為其輸出。

```
{
              "RequestId": "e32d0b83-25f7-41e7-8c8b-e89326ecc87f",
    "SuccessfulList": [
    {
            "Id": "78b45deb-1163-46b1-879b-7d8fcbe9d9d6",
            "Name": "session1",
            "Owner": "user1",
```

```
            "Type": "CONSOLE",
            "State": "CREATING"
    },
    {
            "Id": " a0c743c4-9ff7-43ce-b13f-0c4d55a268dd",
            "Name": "session2",
            "Owner": "user1",
            "Type": "VIRTUAL",
            "State": "CREATING"
    },
    {
            "Id": " 10311636-df90-4cd1-bcf7-474e9675b7cd",
            "Name": "session3",
            "Owner": "user1",
            "Type": "VIRTUAL",
            "State": "CREATING"
    }
    ],
    "UnsuccessfulList": [
    ]
}
```

DescribeServers

說明一或多個 NICE DCV 伺服器。

主題

- 請求參數
- 回應參數
- 範例

請求參數

`ServerIds`

要描述的 NICE DCV DCV 伺服器的識別碼。如果未指定 ID，則所有伺服器都會以分頁輸出傳回。

類型：字串陣列

必要：否

NextToken

擷取下一頁結果使用的字符。

類型：字串

必要：否

MaxResults

分頁輸出中要求傳回的結果數目上限。使用此參數時，請求只返回指定數量的結果與NextToken響應元素一起在單個頁面中。初始請求的剩餘結果可以通過發送帶有返回NextToken值的另一個請求來查看。

有效範圍：1 到 1000

預設：1000

類型：整數

必要：否

回應參數

RequestId

要求的唯一 ID。

Servers

有關 NICE DCV 伺服器的資訊。此資料結構包含下列巢狀回應參數：

Id

NICE DCV DCV 伺服器的唯一 ID。

Ip

NICE DCV DCV 伺服器的 IP 地址。

Hostname

NICE DCV 伺服器的主機名稱。

Endpoints

有關 NICE DCV 伺服器端點的資訊。此資料結構包含下列巢狀回應參數：

IpAddress

伺服器端點的 IP 地址。

Port

伺服器端點的連接埠。

Protocol

伺服器端點使用的通訊協定。可能的值包括：

- HTTP— 端點使用 WebSocket (TCP) 通訊協定。
- QUIC— 端點使用 QUIC (UDP) 通訊協定。

WebUrlPath

伺服器端點的 Web URL 路徑。僅適用於 HTTP 通訊協定。

Version

NICE DCV DCV 伺服器的版本。

SessionManagerAgentVersion

NICE DCV 伺服器上執行的階段作業管理員代理程式版本。

Availability

NICE DCV DCV 服務器的可用性。可能的值包括：

- AVAILABLE— 伺服器可供使用且可以放置工作階段。
- UNAVAILABLE— 伺服器無法使用且無法接受工作階段放置。

UnavailabilityReason

NICE DCV 伺服器無法使用的原因。可能的值包括：

- SERVER_FULL— NICE DCV 伺服器已達到可執行的並行工作階段數目上限。
- SERVER_CLOSED— NICE DCV 伺服器已使用 CloseServerAPI 無法使用。
- UNREACHABLE_AGENT— 工作階段管理員代理人無法與 NICE DCV 伺服器上的工作階段管理員代理程式通訊。

- UNHEALTHY_DCV_SERVER— 工作階段管理員代理程式無法與 NICE DCV 伺服器通訊。
- EXISTING_LOGGED_IN_USER— (僅限視窗 NICE DCV 伺服器) 使用者目前已使用 RDP 登入 NICE DCV 伺服器。
- UNKNOWN— 工作階段管理員代理人無法判斷原因。

ConsoleSessionCount

NICE DCV 伺服器上的主控台工作階段數目。

VirtualSessionCount

NICE DCV 伺服器上的虛擬工作階段數目。

Host

執行 NICE DCV 伺服器之主機伺服器的相關資訊。此資料結構包含下列巢狀回應參數：

Os

有關主機伺服器作業系統的資訊。此資料結構包含下列巢狀回應參數：

Family

作業系統。可能的值包括：

- windows— 主機伺服器正在執行 Windows 作業系統。
- linux— 主機伺服器正在執行 Linux 作業系統。

Name

作業系統的名稱。

Version

作業系統的版本。

KernelVersion

(僅限 Linux) 作業系統的核心版本。

BuildNumber

(僅限視窗) 作業系統的組建編號。

Memory

有關主機伺服器記憶體的資訊。此資料結構包含下列巢狀回應參數：

TotalBytes

主機伺服器上的總記憶體 (以位元組為單位)。

UsedBytes

主機伺服器上已使用的記憶體 (以位元組為單位)。

Swap

有關主機伺服器交換檔的資訊。此資料結構包含下列巢狀回應參數：

TotalBytes

主機伺服器上的交換檔總大小 (以位元組為單位)。

UsedBytes

主機伺服器上使用的交換檔大小 (以位元組為單位)。

Aws

僅適用於在 Amazon EC2 執行個體上執行的 NICE DCV 伺服器。 AWS具體信息。此資料結構包含下列巢狀回應參數：

Region

Amazon EC2 執行個體執行個體的AWS區域。

Ec2InstanceType

Amazon EC2 執行個體類型。

Ec2InstanceId

Amazon EC2 執行個體的唯一 ID。

Ec2ImageId

Amazon EC2 映像檔的識別碼。

CpuInfo

有關主機伺服器 CPU 的資訊。此資料結構包含下列巢狀回應參數：

Vendor

主機伺服器 CPU 的廠商。

ModelName

主機伺服器 CPU 的型號名稱。

Architecture

主機伺服器 CPU 的架構。

NumberOfCpus

主機伺服器上的 CPU 數目。

PhysicalCorePerCpu

每個 CPU 核心數量。

CpuLoadAverage

有關主機伺服器 CPU 負載的資訊。此資料結構包含下列巢狀回應參數：

OneMinute

在過去 1 分鐘期間平均 CPU 負載。

FiveMinutes

在過去 5 分鐘期間平均 CPU 負載。

FifteenMinutes

在過去 15 分鐘期間平均 CPU 負載。

Gpus

主機伺服器 GPU 的相關資訊。此資料結構包含下列巢狀回應參數：

Vendor

主機伺服器 GPU 的廠商。

ModelName

主機伺服器 GPU 的型號名稱。

LoggedInUsers

目前登入主機伺服器的使用者。此資料結構包含下列巢狀回應參數：

Username

登入使用者的使用者名稱。

Tags

已指派給伺服器的標籤。此資料結構包含下列巢狀回應參數：

Key

標籤金鑰。

Value

標籤值。

範例

Python

請求

以下範例說明所有可用的 NICE DCV 伺服器。結果會分頁，每頁顯示兩個結果。

```
from swagger_client.models.describe_servers_request_data import
 DescribeServersRequestData

def get_servers_api():
    api_instance =
 swagger_client.ServersApi(swagger_client.ApiClient(get_client_configuration()))
    set_request_headers(api_instance.api_client)
    return api_instance

def describe_servers(server_ids=None, next_token=None, max_results=None):
    request = DescribeServersRequestData(server_ids=server_ids,
 next_token=next_token, max_results=max_results)
    print('Describe Servers Request:', request)
    api_instance = get_servers_api()
    api_response = api_instance.describe_servers(body=request)
    print('Describe Servers Response', api_response)

def main():
    describe_servers(max_results=2)
```

回應

以下為其輸出。


```
{
    "RequestId": "request-id-123",
    "Servers": [
        {
            "Id": "ServerId123",
            "Ip": "1.1.1.123",
            "Hostname": "node001",
            "DefaultDnsName": "node001",
            "Endpoints": [
                {
                    "IpAddress": "x.x.x.x",
                    "Port": 8443,
                    "WebUrlPath": "/",
                    "Protocol": "HTTP"
                }
            ],
            "Version": "2021.0.10000",
            "SessionManagerAgentVersion": "2021.0.300",
            "Availability": "UNAVAILABLE",
            "UnavailabilityReason": "SERVER_FULL",
            "ConsoleSessionCount": 1,
            "VirtualSessionCount": 0,
            "Host": {
                "Os": {
                    "Family": "windows",
                    "Name": "Windows Server 2016 Datacenter",
                    "Version": "10.0.14393",
                    "BuildNumber": "14393"
                },
                "Memory": {
                    "TotalBytes": 8795672576,
                    "UsedBytes": 1743886336
                },
                "Swap": {
                    "TotalBytes": 0,
                    "UsedBytes": 0
                },
                "Aws": {
                    "Region": "us-west-2b",
                    "EC2InstanceType": "t2.large",
                    "EC2InstanceId": "i-123456789",
                    "EC2ImageId": "ami-12345678987654321"
```

```
                },
                "CpuInfo": {
                    "Vendor": "GenuineIntel",
                    "ModelName": "Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz",
                    "Architecture": "x86_64",
                    "NumberOfCpus": 2,
                    "PhysicalCoresPerCpu": 3
                },
                "CpuLoadAverage": {
                    "OneMinute": 0.04853546,
                    "FiveMinutes": 0.21060601,
                    "FifteenMinutes": 0.18792416
                },
                "Gpus": [],
                "LoggedInUsers": [
                    {
                        "Username": "Administrator"
                    }
                ]
            },
            "Tags": [
                {
                    "Key": "color",
                    "Value": "pink"
                },
                {
                    "Key": "dcv:os-family",
                    "Value": "windows"
                },
                {
                    "Key": "size",
                    "Value": "small"
                },
                {
                    "Key": "dcv:max-virtual-sessions",
                    "Value": "0"
                }
            ]
        },
        {
            "Id": "server-id-12456897",
            "Ip": "1.1.1.145",
            "Hostname": "node002",
            "DefaultDnsName": "node002",
```

```
            "Endpoints": [
                {
                    "IpAddress": "x.x.x.x",
                    "Port": 8443,
                    "WebUrlPath": "/",
                    "Protocol": "HTTP"
                },
                {
                    "IpAddress": "x.x.x.x",
                    "Port": 8443,
                    "Protocol": "QUIC"
                }
            ],
            "Version": "2021.0.10000",
            "SessionManagerAgentVersion": "2021.0.0",
            "Availability": "AVAILABLE",
            "ConsoleSessionCount": 0,
            "VirtualSessionCount": 5,
            "Host": {
                "Os": {
                    "Family": "linux",
                    "Name": "Amazon Linux",
                    "Version": "2",
                    "KernelVersion": "4.14.203-156.332.amzn2.x86_64"
                },
                "Memory": {
                    "TotalBytes": 32144048128,
                    "UsedBytes": 2184925184
                },
                "Swap": {
                    "TotalBytes": 0,
                    "UsedBytes": 0
                },
                "Aws": {
                    "Region": "us-west-2a",
                    "EC2InstanceType": "g3s.xlarge",
                    "EC2InstanceId": "i-123456789",
                    "EC2ImageId": "ami-12345678987654321"
                },
                "CpuInfo": {
                    "Vendor": "GenuineIntel",
                    "ModelName": "Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz",
                    "Architecture": "x86_64",
                    "NumberOfCpus": 4,
```

```
                    "PhysicalCoresPerCpu": 2
                },
                "CpuLoadAverage": {
                    "OneMinute": 2.24,
                    "FiveMinutes": 0.97,
                    "FifteenMinutes": 0.74
                },
                "Gpus": [
                    {
                        "Vendor": "NVIDIA Corporation",
                        "ModelName": "GM204GL [Tesla M60]"
                    }
                ],
                "LoggedInUsers": [
                    {
                        "Username" : "user45687"
                    },
                    {
                        "Username" : "user789"
                    }
                ]
            },
            "Tags": [
                {
                    "Key": "size",
                    "Value": "big"
                },
                {
                    "Key": "dcv:os-family",
                    "Value": "linux"
                },
                {
                    "Key": "dcv:max-virtual-sessions",
                    "Value": "10"
                },
                {
                    "Key": "color",
                    "Value": "blue"
                }
            ]
        }
    ]
}
```

DescribeSessions

說明一或多個 NICE DCV 工作階段。

主題

請求參數

`SessionIds`

要描述的工作階段 ID。

類型：字串

必要：否

`NextToken`

擷取下一頁結果使用的字符。

類型：字串

必要：否

`Filters`

要套用至要求的其他篩選。支援的篩選器包括：

- 標籤：指派給工作階段的標籤。
- 擁有者 — 階段作業擁有者。

類型：字串

必要：否

回應參數

Id

工作階段的唯一 ID。

Name

工作階段的名稱。

Owner

工作階段的擁有者。

Server

有關執行階段作業之伺服器的資訊。此資料結構包含下列巢狀回應參數：

Ip

NICE DCV DCV 伺服器主機的 IP 地址。

Hostname

NICE DCV 伺服器主機的主機名稱。

Port

NICE DCV 伺服器與 NICE DCV 用戶端進行通訊的連接埠。

Endpoints

有關 NICE DCV 伺服器端點的資訊。此資料結構包含下列巢狀回應參數：

IpAddress

伺服器端點的 IP 地址。

Port

伺服器端點的連接埠。

Protocol

伺服器端點使用的通訊協定。可能的值包括：

- HTTP— 端點使用 WebSocket (TCP) 通訊協定。

- QUIC— 端點使用 QUIC (UDP) 通訊協定。

WebUrlPath

伺服器端點的 Web URL 路徑。僅適用於 HTTP 通訊協定。

Tags

已指派給伺服器的標籤。此資料結構包含下列巢狀回應參數：

Key

標籤金鑰。

Value

標籤值。

Type

工作階段的類型。

State

工作階段目前的狀態。可能值為：

- CREATING-代理程式正在建立工作階段。
- READY-工作階段已準備好接受用戶端連線。
- DELETING-正在刪除工作階段段段段段段段。
- DELETED-工作階段段段段段段段已刪除。
- UNKNOWN-無法判斷工作階段的狀態。代理程式和代理程式可能無法通訊。

Substate

階段作業的目前子狀態。可能值為：

- SESSION_PLACING-工作階段正在等待放置在可用的 DCV 伺服器上。
- PENDING_PREPARATION-工作階段已建立但無法使用；連結至 DCV 伺服器。

CreationTime

建立工作階段的日期和時間。

LastDisconnectionTime

上次斷線的日期和時間。

NumOfConnections

作用中用中用戶端連線的數目。

StorageRoot

指定要用於儲存工作階段之資料夾的路徑。如需有關 NICE DCV 工作階段儲存的詳細資訊，請參閱《NICE DCV 管理員指南》中的啟用工作階段儲存。

類型：字串

必要：否

範例

Python

請求

下列範例說明擁有user1且具有標籤的工作階段os=windows。

```
from swagger_client.models.describe_sessions_request_data import
 DescribeSessionsRequestData
from swagger_client.models.key_value_pair import KeyValuePair

def get_sessions_api():
    api_instance =
 swagger_client.SessionsApi(swagger_client.ApiClient(get_client_configuration()))
    set_request_headers(api_instance.api_client)
    return api_instance

def describe_sessions(session_ids=None, next_token=None, tags=None, owner=None):
    filters = list()
    if tags:
        for tag in tags:
            filter_key_value_pair = KeyValuePair(key='tag:' + tag['Key'],
 value=tag['Value'])
            filters.append(filter_key_value_pair)
    if owner:
```

```
        filter_key_value_pair = KeyValuePair(key='owner', value=owner)
        filters.append(filter_key_value_pair)

    request = DescribeSessionsRequestData(session_ids=session_ids, filters=filters,
 next_token=next_token)
    print('Describe Sessions Request:', request)
    api_instance = get_sessions_api()
    api_response = api_instance.describe_sessions(body=request)
    print('Describe Sessions Response', api_response)

def main():
    describe_sessions(
        owner='user1',
        tags=[{'Key': 'os', 'Value': 'windows'}])
```

回應

以下為其輸出。

```
{
    "Sessions": [
        {
            "Id": "SessionId1897",
            "Name": "a session name",
            "Owner": "an owner 1890",
            "Server": {
                "Ip": "1.1.1.123",
                "Hostname": "server hostname",
                "Port": "1222",
                "Endpoints": [
                    {
                        "IpAddress": "x.x.x.x",
                        "Port": 8443,
                        "WebUrlPath": "/",
                        "Protocol": "HTTP"
                    },
                    {
                        "IpAddress": "x.x.x.x",
                        "Port": 9443,
                        "WebUrlPath": "/",
                        "Protocol": "HTTP"
                    },
                    {
                        "IpAddress": "x.x.x.x",
```

```
                    "Port": 8443,
                    "WebUrlPath": "",
                    "Protocol": "QUIC"
                }
            ],
            "Tags": [
                {
                    "Key": "os",
                    "Value": "windows"
                },
                {
                    "Key": "ram",
                    "Value": "4gb"
                }
            ]
        },
        "Type": "VIRTUAL",
        "State": "READY",
        "CreationTime": "2020-10-06T10:15:31.633Z",
        "LastDisconnectionTime": "2020-10-06T10:15:31.633Z",
        "NumOfConnections": 2,
        "StorageRoot" : "/storage/root"
    },
    {
        "Id": "SessionId1895",
        "Name": "a session name",
        "Owner": "an owner 1890",
        "Server": {
            "Ip": "1.1.1.123",
            "Hostname": "server hostname",
            "Port": "1222",
            "Endpoints": [
                {
                    "IpAddress": "x.x.x.x",
                    "Port": 8443,
                    "WebUrlPath": "/",
                    "Protocol": "HTTP"
                },
                {
                    "IpAddress": "x.x.x.x",
                    "Port": 9443,
                    "WebUrlPath": "/",
                    "Protocol": "HTTP"
                },
```

```
                    {
                        "IpAddress": "x.x.x.x",
                        "Port": 8443,
                        "WebUrlPath": "",
                        "Protocol": "QUIC"
                    }
                ],
                "Tags": [
                    {
                        "Key": "os",
                        "Value": "windows"
                    },
                    {
                        "Key": "ram",
                        "Value": "4gb"
                    }
                ]
            },
            "Type": "VIRTUAL",
            "State": "DELETING",
            "CreationTime": "2020-10-06T10:15:31.633Z",
            "LastDisconnectionTime": "2020-10-06T10:15:31.633Z",
            "NumOfConnections": 2,
            "StorageRoot" : "/storage/root"
        }
    ]
}
```

DeleteSessions

刪除指定的 NICE DCV 工作階段，並將其從代理程式的快取中移除。

主題

- 請求參數
- 回應參數
- 範例

請求參數

SessionId

欲刪除的工作階段 ID。

類型：字串

必要：是

Owner

欲刪除的工作階段擁有者。

類型：字串

必要：是

Force

嘗試從 NICE DCV 伺服器中刪除工作階段，從代理程式的快取中移除工作階段。這對於從 Broker 的快取中移除過期的工作階段非常有用。例如，如果 NICE DCV 伺服器已停止，但工作階段仍在 Broker 上註冊，請使用此旗標從 Broker 的快取中清除工作階段。

請記住，如果工作階段仍處於作用中狀態，則會由 Broker 重新快取。

有效值：`true | false`

類型：布林值

必要：否

回應參數

SessionId

工作階段的 ID

State

只有在成功刪除工作階段時才傳回。指示工作階段段段段段段段段段段段段段 如果要求順利完成，工作階段會轉換為DELETING狀態。刪除工作階段段段段段段段段可能需要幾分鐘。當它被刪除時，狀態會從轉換DELETING為DELETED。

FailureReason

只有在無法刪除某些工作階段時才傳回。指出無法刪除階段作業的原因。

範例

Python

請求

下列範例會刪除兩個工作階段 — 一個 ID 為其擁SessionId123有者的工作階段user1，而 ID 為的工作階段則由所擁有user99。SessionIdabc

```
from swagger_client.models.delete_session_request_data import
 DeleteSessionRequestData

def get_sessions_api():
    api_instance =
 swagger_client.SessionsApi(swagger_client.ApiClient(get_client_configuration()))
    set_request_headers(api_instance.api_client)
    return api_instance

def delete_sessions(sessions_to_delete, force=False):
    delete_sessions_request = list()
    for session_id, owner in sessions_to_delete:
        a_request = DeleteSessionRequestData(session_id=session_id, owner=owner,
 force=force)
        delete_sessions_request.append(a_request)

    print('Delete Sessions Request:', delete_sessions_request)
    api_instance = get_sessions_api()
    api_response = api_instance.delete_sessions(body=delete_sessions_request)
    print('Delete Sessions Response', api_response)

def main():
    delete_sessions([('SessionId123', 'an owner user1'), ('SessionIdabc',
 'user99')])
```

回應

以下為其輸出。 SessionId123已成功刪除，但SessionIdabc無法刪除。

```
{
    "RequestId": "10311636-df90-4cd1-bcf7-474e9675b7cd",
    "SuccessfulList": [
        {
            "SessionId": "SessionId123",
            "State": "DELETING"
        }
    ],
    "UnsuccessfulList": [
        {
            "SessionId": "SessionIdabc",
            "FailureReason": "The requested dcvSession does not exist"
        }
    ]
}
```

GetSessionConnectionData

取得特定使用者連線至特定 NICE DCV 工作階段的連線資訊。

主題

請求參數

`SessionId`

要檢視其連線資訊工作階段段段段段段的工作階段 ID。

類型：字串

必要：是

`User`

要檢視其連線資訊之使用者名稱。

類型：字串

必要：是

回應參數

Id

工作階段的唯一 ID。

Name

工作階段的名稱。

Owner

工作階段的擁有者。

Server

有關執行階段作業之伺服器的資訊。此資料結構包含下列巢狀回應參數：

Ip

NICE DCV 伺服器主機的 IP 地址。

Hostname

NICE DCV 伺服器主機的主機名稱。

Port

NICE DCV 伺服器與 NICE DCV 用戶端進行通訊的連接埠。

Endpoints

有關 NICE DCV 伺服器端點的資訊。此資料結構包含下列巢狀回應參數：

IpAddress

伺服器端點的 IP 地址。

Port

伺服器端點的連接埠。

Protocol

伺服器端點使用的通訊協定。可能的值包括：

- HTTP— 端點使用 WebSocket (TCP) 通訊協定。
- QUIC— 端點使用 QUIC (UDP) 通訊協定。

WebUrlPath

伺服器端點的 Web URL 路徑。僅適用於 HTTP 通訊協定。

WebUrlPath

NICE DCV 伺服器組態檔案的路徑。

Tags

已指派給伺服器的標籤。此資料結構包含下列巢狀回應參數：

Key

標籤金鑰。

Value

標籤值。

Type

工作階段的類型。

State

工作階段目前的狀態。可能值為：

- CREATING-代理程式正在建立工作階段。
- READY-工作階段已準備好接受用戶端連線。
- DELETING-正在刪除工作階段段段段段段。
- DELETED-工作階段段段段段段段已刪除。
- UNKNOWN-無法判斷工作階段的狀態。代理程式和代理程式可能無法通訊。

CreationTime

建立工作階段的日期和時間。

LastDisconnectionTime

上次斷線的日期和時間。

NumOfConnections

使用者與工作階段的同時連線數目。

ConnectionToken

用於連接到會話的身份驗證令牌。

附加信息

從此 API 獲得的信息可以傳遞給 NICE DCV 客戶端，以便連接到 NICE DCV 會話。

在 NICE DCV Web 客戶端的情況下，您可以構建一個可以在瀏覽器中打開的 URL。網址具有以下格式：

```
https://{Ip}:{Port}{WebUrlPath}?authToken={ConnectionToken}#{SessionId}.
```

在 NICE DCV 原生用戶端的情況下，您可以使用dcv://結構描述建立 URL。安裝 NICE DCV 原生用戶端時，它會將自己註冊到系統中，做為dcv:// URL 的處理常式。網址具有以下格式：

```
dcv://{Ip}:{Port}{WebUrlPath}?authToken={ConnectionToken}#{SessionId}.
```

> **Note**
>
> 如果您使用的是 Amazon EC2，則 IP 地址應該是公共的 IP 地址。如果您的組態在閘道後方有 NICE DCV 主機，請指定閘道位址，而不是 SessionConnectionData API 傳回的位址。

範例

Python

請求

下列範例會取得使用者名稱為的使用者的連線資訊，以user1及 ID 為的工作階段sessionId12345。

```
def get_session_connection_api():
    api_instance =
 swagger_client.GetSessionConnectionDataApi(swagger_client.ApiClient(get_client_configuratio
    set_request_headers(api_instance.api_client)
    return api_instance


def get_url_to_connect(api_response):
    ip_address = api_response.session.server.ip
    port = api_response.session.server.port
    web_url_path = api_response.session.server.web_url_path
    connection_token = api_response.connection_token
    session_id = api_response.session.id
    url = f'https://{ip_address}:{port}{web_url_path}?
authToken={connection_token}#{session_id}'
    return url


def get_session_connection_data(session_id, user):
    api_response =
 get_session_connection_api().get_session_connection_data(session_id=session_id,
 user=user)
    url_to_connect = get_url_to_connect(api_response)
    print('Get Session Connection Data Response:', api_response)
    print('URL to connect: ', url_to_connect)


def main():
    get_session_connection_data('sessionId12345', 'user1')
```

回應

以下為其輸出。

```
{
    "Session": {
        "Id": "sessionId12345",
        "Name": "a session name",
        "Owner": "an owner 1890",
        "Server": {
```

```
            "Ip": "1.1.1.123",
            "Hostname": "server hostname",
            "Port": "1222",
            "endpoints": [
                {
                    "port": 8443,
                    "web_url_path": "/",
                    "protocol": "HTTP"
                },
                {
                    "port": 9443,
                    "web_url_path": "/",
                    "protocol": "HTTP"
                },
                {
                    "port": 8443,
                    "web_url_path": "",
                    "protocol": "QUIC"
                }
            ],
            "WebUrlPath": "/path",
            "Tags": [
                {
                    "Key": "os",
                    "Value": "windows"
                },
                {
                    "Key": "ram",
                    "Value": "4gb"
                }
            ]
        },
        "Type": "VIRTUAL",
        "State": "UNKNOWN",
        "CreationTime": "2020-10-06T10:15:31.633Z",
        "LastDisconnectionTime": "2020-10-06T10:15:31.633Z",
        "NumOfConnections": 2
    },
    "ConnectionToken":
 "EXAMPLEiOiJmOWM1YTRhZi1jZmU0LTQ0ZjEtYjZlOC04ZjY0YjM4ZTE2ZDkiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUz
tngiKXevUxhhJvm3BPJYRs9NPE4GCJRTc13EXAMPLEIxNEPPh5IMcVmROfU1WKPnry4ypPTp3rsZ7YWjCTSfs1GoN3R_
Kqtpd5GH0D-E8FwsedV-
Q2bRQ4y9y1q0MgFU4QjaSMypUuYR0YjkCaoainjmEZew4A33fG40wATrBvoivBiNWdNpytHX2CDOuk_k0k_DWeZjMvv9
h_GaMgHmltqBIA4jdPD7i0CmC2e7413KFy-
```

```
EQ4Ej1cM7RjLwhFuWpKWAVJxogJjYpfoKKaPo4KxvJjJIPYhkscklINQpe2W5rnlxCq7sC7ptcGw17DUobP7egRv9H37
hK1G4G8erHvl9HIrTR9_c884fNrTCC8DvC062e4KYdLkAhhJmboN9CAGIGFyd2c1AY_CzzvDL0EXAMLE"
}
```

GetSessionScreenshots

獲取一個或多個 NICE DCV 會話的屏幕截圖。

螢幕擷取畫面的影像檔類型和解析度取決於工作階段管理員中介組態。若要修改影像檔案類型，請設定`session-screenshot-format`參數。若要修改解析度，請設定`session-screenshot-max-width`和`session-screenshot-max-height`參數。如需詳細資訊，請參閱《NICE DCV 工作階段管理員管理員指南》中的代理人組態檔。

主題

- 請求參數
- 回應參數
- 範例

請求參數

`SessionId`

要從中取得螢幕擷取畫面的 NICE DCV 工作階段識別碼。

類型：字串

必要：是

回應參數

`RequestId`

要求的唯一 ID。

`SuccessfulList`

有關成功截圖的信息。此資料結構包含下列巢狀回應參數：

SessionScreenshot

有關屏幕截圖的信息。此資料結構包含下列巢狀回應參數：

SessionId

從中擷取螢幕擷取畫面的 NICE DCV 工作階段識別碼。

Images

關於影像的資訊。此資料結構包含下列巢狀回應參數：

Format

圖像的格式。可能的值包括：jpeg和png。

Data

截圖圖像 base64 編碼格式。

CreationTime

擷取螢幕擷取畫面擷取的日期和時間。

Primary

指出螢幕擷取畫面是否為 NICE DCV 工作階段的主要顯示器。

UnsuccessfulList

有關不成功的屏幕截圖信息。此資料結構包含下列巢狀回應參數：

GetSesionScreenshotRequestData

失敗的原始要求。

SessionId

要從中擷取螢幕擷取畫面的 NICE DCV 工作階段識別碼。

FailureReason

失敗的原因。

範例

Python

請求

下列範例會從兩個工作階段 (sessionId1和sessionId2) 取得螢幕擷取畫面。工作階段sessionId2不存在，導致失敗。

```
from swagger_client.models.describe_servers_request_data import
 DescribeServersRequestData

def get_sessions_api():
    api_instance =
 swagger_client.ServersApi(swagger_client.ApiClient(get_client_configuration()))
    set_request_headers(api_instance.api_client)
    return api_instance

def get_session_screenshots(session_ids):
    request = [GetSessionScreenshotRequestData(session_id=session_id) for session_id
 in session_ids]
    print('Get Session Screenshots Request:', request)
    api_instance = get_sessions_api()
    api_response = api_instance.get_session_screenshots(body=request)
    print('Get Session Screenshots Response:', api_response)

def main():
    get_session_screenshots(["sessionId1", "sessionId2"])
```

回應

以下為其輸出。

```
{
    "RequestId": "542735ef-f6ab-47d8-90e5-23df31d8d166",
    "SuccessfulList": [
        {
            "SessionScreenshot": {
                "SessionId": "sessionId1",
                "Images": [
                    {
                        "Format": "png",
                        "Data": "iVBORw0KGgoAAAANSUhEUgAAAEXAMPLE",
                        "CreationTime": "2021-03-30T15:47:06.822Z",
                        "Primary": true
                    }
                ]
            }
        }
```

```
    ],
    "UnsuccessfulList": [
        {
            "GetSessionScreenshotRequestData": {
                "SessionId": "sessionId2"
            },
            "FailureReason": "Dcv session not found."
        }
    ]
}
```

OpenServers

開啟一或多個 NICE DCV 伺服器。在 NICE DCV 伺服器上建立 NICE DCV 工作階段之前，您必須將伺服器的狀態變更為開啟。NICE DCV 伺服器開啟後，您可以在伺服器上建立 NICE DCV 工作階段。

主題

- 請求參數
- 回應參數
- 範例

請求參數

`ServerId`

欲開啟的伺服器 ID。

類型：字串

必要：是

回應參數

`RequestId`

要求的唯一 ID。

SuccessfulList

已成功開啟的 NICE DCV 伺服器的相關資訊。此資料結構包含下列巢狀回應參數：

ServerId

已成功開啟之伺服器的識別碼。

UnsuccessfulList

無法開啟的 NICE DCV 伺服器的相關資訊。此資料結構包含下列巢狀回應參數：

OpenServerRequestData

原始要求失敗的相關資訊。此資料結構包含下列巢狀回應參數：

ServerId

無法開啟的 NICE DCV 伺服器的識別碼。

FailureCode

失敗的代碼。

FailureReason

失敗的原因。

範例

Python

請求

下列範例會開啟兩個 NICE DCV 伺服器 (`serverId1`和`serverId2`)。

```
from swagger_client.models import OpenServerRequestData


def get_servers_api():
    api_instance =
 swagger_client.ServersApi(swagger_client.ApiClient(get_client_configuration()))
    set_request_headers(api_instance.api_client)
    return api_instance

def open_servers(server_ids):
```

```
    request = [OpenServerRequestData(server_id=server_id) for server_id in
 server_ids]
    print('Open Servers Request:', request)
    api_instance = get_servers_api()
    api_response = api_instance.open_servers(body=request)
    print('Open Servers Response:', api_response)

def main():
    open_servers(["serverId1", "serverId2"])
```

回應

以下為其輸出。

```
{
    "RequestId": "1e64830f-0a27-41bf-8147-0f3411791b64",
    "SuccessfulList": [
        {
            "ServerId": "serverId1"
        }
    ],
    "UnsuccessfulList": [
        {
            "OpenServerRequestData": {
                "ServerId": "serverId2"
            },
            "FailureCode": "DCV_SERVER_NOT_FOUND",
            "FailureReason": "Dcv server not found."
        }
    ]
}
```

UpdateSessionPermissions

更新特定 NICE DCV 工作階段的使用者權限。

主題

- 請求參數
- 回應參數
- 範例

請求參數

`SessionId`

要變更其使用權限之工作階段段段段段段的工作階段 ID。

類型：字串

必要：是

`Owner`

要變更其權限之工作階段的擁有者。

類型：字串

必要：是

`PermissionFile`

要使用之權限檔案的 Base64 編碼內容。如需詳細資訊，請參閱《NICE DCV 管理員指南》中的 < 設定 NICE DCV 授權 > 。

類型：字串

必要：是

回應參數

`SessionId`

工作階段的 ID。

範例

Python

請求

下列範例會為工作階段 ID 為的工作階段設定新權限`SessionId1897`。

```
from swagger_client.models.update_session_permissions_request_data import
 UpdateSessionPermissionsRequestData
```

```
def get_session_permissions_api():
    api_instance =
 swagger_client.SessionPermissionsApi(swagger_client.ApiClient(get_client_configuration()))
    set_request_headers(api_instance.api_client)
    return api_instancedef
 update_session_permissions(session_permissions_to_update):
    update_session_permissions_request = list()
    for session_id, owner, permissions_base64_encoded in
 session_permissions_to_update:
        a_request = UpdateSessionPermissionsRequestData(
            session_id=session_id, owner=owner,
 permissions_file=permissions_base64_encoded)
        update_session_permissions_request.append(a_request)
    print('Update Session Permissions Request:', update_session_permissions_request)
    api_instance = get_session_permissions_api()
    api_response =
 api_instance.update_session_permissions(body=update_session_permissions_request)
    print('Update Session Permissions Response:', api_response)

def main():
    update_session_permissions([('SessionId1897', 'an owner 1890',
 'file_base64_encoded')])
```

回應

以下為其輸出。

```
{
   'request_id': 'd68ebf66-4022-42b5-ba65-99f89b18c341',
    'successful_list': [
        {'
            session_id': 'SessionId1897'
        }
    ],
    'unsuccessful_list': []
}
```

NICE DCV 工作階段管理員的發行說明和文件歷史記錄

此頁面提供 NICE DCV 工作階段管理員的發行說明和文件歷史記錄。

主題

NICE DCV 工作階段管理員發行說明

本節提供 NICE DCV 工作階段管理員的主要更新、功能版本和錯誤修正的概觀。所有更新都是按發布日期組織的。我們會經常更新文件，以解決您傳送給我們的意見反應。

主題

2023.1— 二二三年十一月九日

建置編號	變更與錯誤修正
• 經紀人： • 代理人: • CLI：140	• 錯誤修正與效能改進

2023.0-15065— 二二二三年五月四日

建置編號	變更與錯誤修正
• 經紀人： • 代理人: • CLI: 132	• 在 ARM 平台上增加了對紅帽企業版 Linux 9，洛奇 Linux 9 和 CentOS 流 9 的支持。

2023.0-14852— 二二三年三月二十八日

建置編號	變更與錯誤修正
• 經紀人： • 代理人: • CLI: 132	• 增加了對紅帽企業版 Linux 9、洛基 Linux 9 和 CentOS 流 9 的支援。

2022.2-13907— 二 ○○ 二年十一月十一日

建置編號	變更與錯誤修正
• 經紀人 • 代理人: • CLI: 123	• 在`DescribeSessions` 響應中添加了一個`Substate`字段。 • 修正可能導致 CLI 無法根據使用中的 URL 連線至代理程式的問題。

2022.1-13067— 二零二二年六月二十九日

建置編號	變更與錯誤修正
• 經紀人 • 代理人: • CLI：114	• 已新增在 AWS Graviton 執行個體上執行代理程式的支援。 • 增加了對 Ubuntu 22.04 的代理程式和代理程式支援。

2022.0-11952— 二零二二年二月二十三日

建置編號	變更與錯誤修正
• 經紀人： • 代理人: • CLI: 112	• 已將記錄輪換功能新增至代理程式。 • 已新增組態參數以在代理程式中設定 Java 本位目錄。 • 改善 Broker 中從快取到磁碟的資料清除功能。 • 在 CLI 中修正了 URL 驗證。

2021.3-11591— 二零二一年十二月二十日

建置編號	新的 功能
• 經紀人：307 • 代理人: • CLI：92	• 新增了與 NICE DCV 連線閘道整合的支援。 • 增加了對 Ubuntu 18.04 和 Ubuntu 20.04 的代理支持。

2021.2-11445— 二零二一年十一月十八日

建置編號	變更與錯誤修正
• 經紀人： • 代理人: • CLI 數：54	• 修正驗證包含 Windows 網域的登入名稱的問題。

2021.2-11190— 二零二一年十月十一日

建置編號	變更與錯誤修正
• 經紀人：254 • 代理人: • CLI 數：54	• 修正命令列介面中無法啟動 Windows 工作階段的問題。

2021.2-11042— 二零二一年九月一日

建置編號	新的 功能	變更與錯誤修正
• 經紀人：254 • 代理人: • CLI：37	• NICE DCV 工作階段管理員現在提供命令列介面 (CLI) 支援。您可以在 CLI 中建立和管理 NICE DCV 工作階段，而不是呼叫 API。 • NICE DCV 會話管理器引入了代理人數據持久性。為了提高可用性，代理程式可以在外部資料存放區上保留伺服器狀態資訊，並在啟動時還原資料。	• 註冊外部授權伺服器時，您現在可以指定授權伺服器用來簽署 JSON 格式的 Web Token 的演算法。透過此變更，您可以使用 Azure AD 做為外部授權伺服器。

2021.1-10557— 二零二一年五月三十一日

建置編號	新的 功能	變更與錯誤修正
• 經紀人：214 • 代理人:365	• NICE DCV 會話管理器添加了對傳遞到 Linux 上自動運行文件的輸入參數的支持。 • 伺服器屬性現在可以根據需求傳遞給 CreateSessionsAPI。	• 我們修復了 Windows 上自動運行文件的問題。

2021.0-10242— 二零二一年四月十二日

建置編號	變更與錯誤修正

• 經紀人：183
• 代理人:
	• NICE DCV DCV 會話管理器引入了以下新 API：
	  • OpenServers
	  • CloseServers
	  • DescribeServers
	  • GetSessionScreenshots
	• 它還引入了以下新的配置參數：
	  • 代理人參數：session-screenshot-max-width session-screenshot-max-height session-screenshot-format、create-sessions-queue-max-size 、、和create-sessions-queue-max-time-seconds 。
	  • 代理程式參數：agent.autorun_folder max_virtual_sessions 、和max_concurrent_sessions_per_user 。
	
	  代理程式參數：agent.autorun_folder max_virtual_sessions 、和max_concurrent_sessions_per_user 。
	
	  代理程式參數：agent.autorun_folder max_virtual_sessions 、和max_concurrent_sessions_per_user 。

2020.2-9662— 二零二零年十二月四日

建置編號	變更與錯誤修正
• 經紀人：114 • 代理人:11	• 我們修正了自動產生的 TLS 憑證的問題，導致 Broker 無法啟動。

2020.2-9508— 二零二零年十一月十一日

建置編號	變更與錯誤修正
• 經紀人：78 • 代理人:	• NICE DCV 工作階段管理員的初始版本。

文件歷史紀錄

下表說明此版本 NICE DCV 工作階段管理員的文件。

變更	描述	日期
NICE DCV 民政署	NICE DCV DCV 會話管理器已經更新了 NICE DCV DCV 2023.1。如需詳細資訊，請參閱2023.1— 二二三年十一月九日。	2023年11月9日
NICE DCV 市民商業	NICE DCV DCV 會話管理器已更新 NICE DCV DCV 2023.0。如需詳細資訊，請參閱2023.0-14852— 二二三年三月二十八日。	2023 年 3 月 28 日
NICE DCV 民政署版本 2022.2	NICE DCV DCV 會話管理器已經更新了 NICE DCV DCV 2022.2。如需詳細資訊，請參閱2022.2-13907— 二 ○○ 二年十一月十一日。	2022 年 11 月 11 日
NICE DCV 市民商業中心 2022.1 版	NICE DCV DCV 會話管理器已更新為 NICE DCV DCV 2022.1。如需詳細資訊，請參閱2022.1-13067— 二零二二年六月二十九日。	2022 年 6 月 29 日
NICE DCV 市民商業中心	NICE DCV DCV 會話管理器已更新 NICE DCV DCV 2022.0。如需詳細資	2022 年 2 月 23 日

變更	描述	日期
	訊，請參閱2022.0-11952— 二零二二年二月二十三日。	
NICE DCV 民政署版本 2021.3	NICE DCV DCV 會話管理器已更新為 NICE DCV DCV 2021.3。如需詳細資訊，請參閱2021.3-11591— 二零二一年十二月二十日。	2021 年 12 月 20 日
NICE DCV 民政署版本 2021.2	NICE DCV DCV 會話管理器已經更新了 NICE DCV DCV 2021.2。如需詳細資訊，請參閱2021.2-11042— 二零二一年九月一日。	2021年9月01 日
NICE DCV 市民商業中心	NICE DCV DCV 會話管理器已更新為 NICE DCV DCV 2021.1。如需詳細資訊，請參閱2021.1-10557— 二零二一年五月三十一日。	2021年5月31日
NICE DCV 民政署版本	NICE DCV DCV 會話管理器已更新 NICE DCV DCV 2021.0。如需詳細資訊，請參閱2021.0-10242— 二零二一年四月十二日。	2021 年 4 月 12 日
NICE DCV 會話管理器的初始版本	此內容的第一次出版。	2020 年 11 月 11 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。