aws

開發人員指南

Amazon DCV Session Manager

Amazon DCV Session Manager: 開發人員指南

Table of Contents

什麼是 Amazon DCV Session Manager？

> **Note**
>
> Amazon DCV 先前稱為 NICE DCV。

Amazon DCV Session Manager 是一組可安裝的軟體套件（代理程式和代理程式）和應用程式程式設計界面 (API)，可讓開發人員和獨立軟體供應商 (ISVs) 輕鬆地建置前端應用程式，以程式設計方式跨 Amazon DCV 伺服器機群建立和管理 Amazon DCV 工作階段的生命週期。

本指南說明如何使用 Session Manager APIs 來管理 Amazon DCV 工作階段的生命週期。如需如何安裝和設定 Session Manager Broker 和 Agents 的詳細資訊，請參閱 Amazon DCV Session Manager 管理員指南。

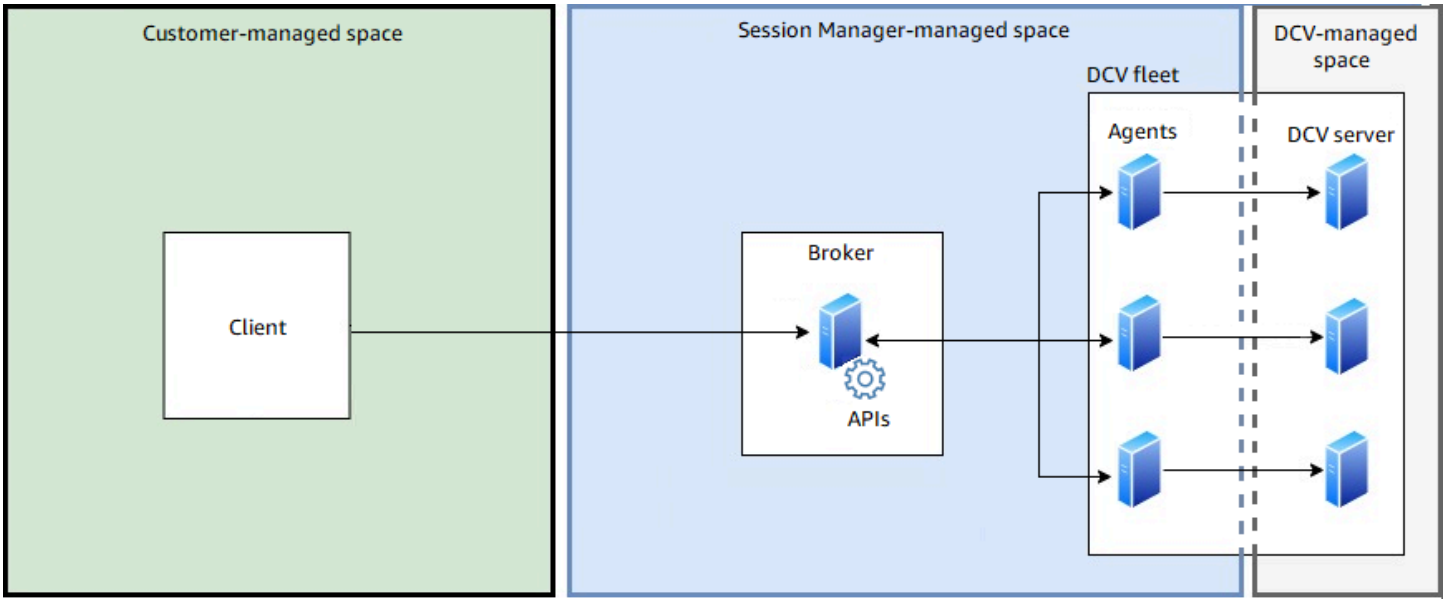
先決條件

開始使用 Session Manager APIs之前，請確定您已熟悉 Amazon DCV 和 Amazon DCV 工作階段。如需詳細資訊，請參閱 Amazon DCV 管理員指南。

主題

Session Manager 的運作方式

下圖顯示 Session Manager 的高階元件。



中介裝置

代理程式是託管和公開 Session Manager APIs的 Web 伺服器。它會接收並處理 API 請求，以從用戶端管理 Amazon DCV 工作階段，然後將指示傳遞給相關的客服人員。代理程式必須安裝在與您的 Amazon DCV 伺服器分開的主機上，但用戶端必須可存取，而且必須能夠存取代理程式。

代理程式

代理程式安裝在機群中的每個 Amazon DCV 伺服器。代理程式會收到代理程式的指示，並在各自的 Amazon DCV 伺服器上執行這些指示。代理程式也會監控 Amazon DCV 伺服器的狀態，並將定期狀態更新傳回給代理程式。

API

Session Manager 會公開一組 REST 應用程式程式設計介面 APIs)，可用於管理 Amazon DCV 伺服器機群上的 Amazon DCV 工作階段。APIs 託管在 上，並由代理程式公開。開發人員可以建立呼叫 APIs的自訂工作階段管理用戶端。

用戶端

用戶端是您開發的前端應用程式或入口網站，用於呼叫代理程式公開的 Session Manager APIs。最終使用者使用 用戶端來管理機群中 Amazon DCV 伺服器上託管的工作階段。

存取字符

若要提出 API 請求，您必須提供存取權杖。註冊的用戶端 APIs 可以從代理程式或外部授權伺服器請求權杖。若要請求和存取字符，用戶端 API 必須提供有效的登入資料。

用戶端 API

用戶端 API 是使用 Swagger Codegen 從 Session Manager API 定義 YAML 檔案產生。用戶端 API 用於提出 API 請求。

Amazon DCV 工作階段

Amazon DCV 工作階段是 Amazon DCV 伺服器能夠接受來自用戶端連線的一段時間。在您的用戶端可以連線至 Amazon DCV 工作階段之前，您必須在 Amazon DCV 伺服器上建立 Amazon DCV 工作階段。Amazon DCV 同時支援主控台和虛擬工作階段，每個工作階段都有指定的擁有者和一組許可。您可以使用 Session Manager APIs 來管理 Amazon DCV 工作階段的生命週期。Amazon DCV 工作階段可以處於下列其中一種狀態：

- 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。
- 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 提供下列功能：

- 提供 Amazon DCV 工作階段資訊 — 取得在多個 Amazon DCV 伺服器上執行之工作階段的相關資訊。
- 管理多個 Amazon DCV 工作階段的生命週期 - 透過一個 API 請求，為多個 Amazon DCV 伺服器中的多個使用者建立或刪除多個工作階段。
- 支援標籤 - 在建立工作階段時，使用自訂標籤將 Amazon DCV 伺服器群組設為目標。
- 管理多個 Amazon DCV 工作階段的許可 - 使用一個 API 請求修改多個工作階段的使用者許可。
- 提供連線資訊 - 擷取 Amazon DCV 工作階段的用戶端連線資訊。
- 支援雲端和內部部署 - 在 、 AWS內部部署上使用 Session Manager，或搭配替代的雲端伺服器。

Session Manager API 入門

Amazon DCV Session Manager API 提供自動化界面，用於管理遠端桌面工作階段。透過此 API，開發人員可以透過程式設計方式建立、列出、啟動、停止和以其他方式控制 DCV 工作階段。這允許將 Amazon DCV 功能整合到自訂應用程式和工作流程。透過利用此 API，組織可以簡化遠端視覺化工作負載的管理，並自動化許多常見的任務。

在開始呼叫 Amazon DCV API 之前，您需要取得可驗證應用程式並授權其存取必要資源的存取字符。Amazon DCV API 使用 OAuth 2.0 進行身分驗證，因此您需要註冊應用程式並擷取必要的登入資料。取得存取權杖後，您就可以開始傳送請求至 Amazon DCV API 端點，以開始處理資料。

主題

步驟 1：產生您的 API 用戶端

Session Manager APIs 是在單一 YAML 檔案中定義。APIs 是以 OpenAPI3.0 規格為基礎，此規格定義了 RESTful APIs 的標準、不依賴語言的界面。如需詳細資訊，請參閱 OpenAPI 規格。

您可以使用 YAML 檔案，以其中一種支援的語言產生 API 用戶端。若要這麼做，您必須使用 Swagger Codegen 3.0 或更新版本。如需支援語言的詳細資訊，請參閱 swagger-codegen 儲存庫。

產生 API 用戶端

1. 從 Session Manager Broker 下載 Session Manager API YAML 檔案。YAML 檔案可在下列 URL 取得。

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

2. 安裝 Swagger Codegen。

   - 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
   ```

步驟 2：註冊您的用戶端 API

API 請求使用存取字符來驗證您的登入資料。這些登入資料是以用戶端向代理程式註冊時產生的用戶端 ID 和用戶端密碼為基礎。

若要存取此字符，您需要向代理程式註冊。使用 register-api-client 註冊用戶端 API。

如果您沒有用戶端的用戶端 ID 和用戶端密碼，您必須向代理程式管理員請求這些 ID 和用戶端密碼。

步驟 3：取得存取權杖並提出 API 請求

此範例會逐步解說設定存取字符的步驟，然後示範如何提出基本 API 請求。這將為您提供基礎知識，以開始建置由 Amazon DCV API 提供支援的更進階應用程式。

在此範例中，我們將示範如何使用 `DescribeSessions` API 來執行此操作。

Example

首先，我們匯入應用程式所需的模型。

然後，我們會宣告用戶端 ID (`__CLIENT_ID`)、用戶端密碼 (`__CLIENT_SECRET`) 和代理程式 URL 的變數，包括連接埠號碼 (`__PROTOCOL_HOST_PORT`)。

接下來，我們會建立名為 的函數build_client_credentials，以產生用戶端登入資料。若要產生用戶端憑證，您必須先將用戶端 ID 和用戶端密碼串連，並將值與冒號 (*client_ID*:*client_password*) 分隔，然後 Base64 編碼整個字串。

```
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')
```

現在我們有用戶端登入資料，我們可以用它向代理程式請求存取權杖。若要執行此操作，我們會建立名為 的函數get_access_token。您必須在 POST上呼叫 https://*Broker_IP*:*8443*/oauth2/token?grant_type=client_credentials，並提供授權標頭，其中包含基本編碼的用戶端登入資料，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，我們建立執行個體化用戶端 API 所需的函數。若要執行個體化用戶端 API，您必須指定用戶端組態和要用於請求的標頭。get_client_configuration 函數會建立組態物件，其中包含代理程式的 IP 地址和連接埠，以及代理程式自我簽署憑證的路徑，您應該已從代理程式管理員收到。set_request_headers 函數會建立請求標頭物件，其中包含用戶端登入資料和存取權杖。

```
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，我們會建立呼叫 DescribeSessions API 的主要方法。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

此參考提供有關可用 API 動作、必要參數和回應格式的詳細資訊，讓您能夠有效地在自己的系統中利用 Session Manager API。使用 Session Manager API，您可以啟動、停止和取得互動式工作階段的詳細資訊。這可讓您將功能自動化並整合到您的應用程式和工作流程。

主題

CloseServers

關閉一或多個 Amazon DCV 伺服器。當您關閉 Amazon DCV 伺服器時，您會讓它無法用於 Amazon DCV 工作階段放置。您無法在已關閉的伺服器上建立 Amazon DCV 工作階段。關閉伺服器可確保其中沒有任何工作階段正在執行，且使用者無法在其中建立新的工作階段。

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`，即使伺服器有執行中的工作階段，也會關閉。工作階段會繼續執行。

類型：布林值

必要：否

回應參數

RequestId

請求的唯一 ID。

SuccessfulList

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

ServerId

已成功關閉的伺服器 ID。

UnsuccessfulList

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

CloseServerRequestData

失敗的原始請求的相關資訊。此資料結構包含下列巢狀回應參數：

ServerId

無法關閉的 Amazon DCV 伺服器 ID。

Force

請求的力參數。

FailureCode

失敗的程式碼。

FailureReason

失敗的原因。

範例

Python

請求

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

```
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

以下是範例輸出。

```
{
    "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的 Amazon DCV 工作階段。

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。這必須是目標 Amazon 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。如需工作階段類型的詳細資訊，請參閱《Amazon DCV 管理員指南》中的 Amazon DCV 工作階段簡介。

有效值：CONSOLE | VIRTUAL

類型：字串

必要：是

InitFile

Linux Amazon DCV 伺服器上的虛擬工作階段支援。Windows 和 Linux Amazon DCV 伺服器上的主控台工作階段不支援此功能。建立工作階段時，要在 Amazon DCV 伺服器上執行以初始化工作階段的自訂指令碼路徑。檔案路徑是相對於為 `agent.init_folder` Agent 組態參數指定的 init 目錄。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《Amazon DCV Session Manager 管理員指南》中的客服人員組態檔案。

類型：字串

必要：否

MaxConcurrents

並行 Amazon DCV 用戶端的數量上限。

類型：整數

必要：否

DcvGlEnabled

指出虛擬工作階段是否設定為使用硬體型 OpenGL。僅支援虛擬工作階段。Windows Amazon DCV 伺服器不支援此參數。

有效值：true | false

類型：布林值

必要：否

PermissionsFile

許可檔案的 Base64-encoded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《Amazon DCV 管理員指南》中的設定 Amazon DCV 授權。

類型：字串

必要：否

EnqueueRequest

指示是否在無法立即完成請求時佇列。

類型：布林值

預設：false

必要：否

AutorunFile

在 Windows Amazon DCV 伺服器上支援主控台工作階段，在 Linux Amazon DCV 伺服器上支援虛擬工作階段。Linux Amazon DCV 伺服器上的主控台工作階段不支援此功能。

要在工作階段中執行之主機伺服器上檔案的路徑。檔案路徑與為 `agent.autorun_folder` Agent 組態參數指定的自動執行目錄相關。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《Amazon DCV Session Manager 管理員指南》中的客服人員組態檔案。

檔案是代表指定的擁有者執行。指定的擁有者必須具有在伺服器上執行 檔案的許可。在 Windows Amazon DCV 伺服器上，檔案會在擁有者登入工作階段時執行。在 Linux Amazon DCV 伺服器上，檔案會在工作階段建立時執行。

類型：字串

必要：否

AutorunFileArguments

Linux Amazon DCV 伺服器上的虛擬工作階段支援。在 Windows 和 Linux Amazon DCV 伺服器上的主控台工作階段中不支援此功能。命令列引數在工作階段內執行時傳遞至 AutorunFile。引數會依其出現在指定陣列中的順序傳遞。可以設定允許的引數數目上限和每個引數允許的長度上限。如需詳細資訊，請參閱《Amazon DCV Session Manager 管理員指南》中的代理程式組態檔案。

類型：字串陣列

必要：否

DisableRetryOnFailure

指出在 Amazon DCV 主機上因任何原因失敗之後，是否不重試建立工作階段請求。如需建立工作階段重試機制的詳細資訊，請參閱《Amazon DCV Session Manager 管理員指南》中的代理程式組態檔案。

類型：布林值

預設：false

必要：否

Requirements

伺服器必須符合的要求，才能放置工作階段。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 DescribeServers API 來擷取。

需求條件表達式：

- *a* ！= *b* 如果 *a* 不等於 *b* 則為 true
- *a* = *b* 如果 *a* 等於 *b*，則為 true
- *a* > *b* 如果 *a* 大於 *b* 則為 true
- *a* >= *b* 如果 *a* 大於或等於 *b* 則為 true
- *##* *a* 小於 *b*，*##* < *b* true
- *##* *a* 小於或等於 *b*，a <= *b* true
- *a* = *b* 如果 *a* 包含字串 *b*，則為 true

布林值運算子需求：

- *a* 和 *b* 若 *a* 和 *b* 為 true
- 如果 *a* 或 *b* 為 true，*##* *a* 或 *b* true
- 如果 為 false#則不是 true

標籤索引鍵必須以 開頭`tag:`，伺服器屬性必須以 開頭`server:`。要求表達式支援括號 ()。

需求範例：

- `tag:color = 'pink' and (server:Host.Os.Family = 'windows' or tag:color := 'red')`

- "server:Host.Aws.Ec2InstanceType := 't2' and server:Host.CpuInfo.NumberOfCpus >= 2"

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

支援的伺服器屬性為：

- 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 Amazon DCV 工作階段儲存體的詳細資訊，請參閱《Amazon DCV 管理員指南》中的啟用工作階段儲存體。

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。如果請求成功完成，工作階段會進入 CREATING 狀態。

Substate

工作階段的子狀態。如果請求成功完成，則子狀態會進入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

以下是範例輸出。

```
{
              "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

描述一或多個 Amazon 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 Amazon DCV 伺服器的 IDs。如果未指定 IDs，則會在分頁輸出中傳回所有伺服器。

類型：字串陣列

必要：否

NextToken

用來擷取結果下一頁的字符。

類型：字串

必要：否

MaxResults

分頁輸出中請求要傳回的結果數目上限。使用此參數時，請求只會傳回單一頁面中指定數量的結果，以及NextToken回應元素。傳送另一個具有傳回NextToken值的請求，即可查看初始請求的剩餘結果。

有效範圍：1 到 1000

預設：1000

類型：整數

必要：否

回應參數

RequestId

請求的唯一 ID。

Servers

Amazon DCV 伺服器的相關資訊。此資料結構包含下列巢狀回應參數：

Id

Amazon DCV 伺服器的唯一 ID。

Ip

Amazon DCV 伺服器的 IP 地址。

Hostname

Amazon DCV 伺服器的主機名稱。

Endpoints

Amazon DCV 伺服器端點的相關資訊。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

伺服器端點所使用的通訊協定。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

Amazon DCV 伺服器的版本。

SessionManagerAgentVersion

在 Amazon DCV 伺服器上執行的 Session Manager Agent 版本。

Availability

Amazon DCV 伺服器的可用性。可能的值包括：

- AVAILABLE — 伺服器已可使用並準備好放置工作階段。
- UNAVAILABLE — 伺服器無法使用，無法接受工作階段置放。

UnavailabilityReason

Amazon DCV 伺服器無法使用的原因。可能的值包括：

- SERVER_FULL — Amazon DCV 伺服器已達到可執行的並行工作階段數量上限。
- SERVER_CLOSED — Amazon DCV 伺服器已使用 CloseServer API 無法使用。
- UNREACHABLE_AGENT — Session Manager Broker 無法與 Amazon DCV 伺服器上的 Session Manager Agent 通訊。
- UNHEALTHY_DCV_SERVER — Session Manager Agent 無法與 Amazon DCV 伺服器通訊。
- EXISTING_LOGGED_IN_USER — （僅限 Windows Amazon DCV 伺服器）使用者目前使用 RDP 登入 Amazon DCV 伺服器。
- UNKNOWN — Session Manager Broker 無法判斷原因。

ConsoleSessionCount

Amazon DCV 伺服器上的主控台工作階段數目。

VirtualSessionCount

Amazon DCV 伺服器上的虛擬工作階段數量。

Host

執行 Amazon DCV 伺服器的主機伺服器相關資訊。此資料結構包含下列巢狀回應參數：

Os

主機伺服器作業系統的相關資訊。此資料結構包含下列巢狀回應參數：

Family

作業系統系列。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 Windows) 作業系統的建置編號。

Memory

主機伺服器的記憶體相關資訊。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

主機伺服器上使用過的記憶體，以位元組為單位。

Swap

主機伺服器交換檔案的相關資訊。此資料結構包含下列巢狀回應參數：

TotalBytes

主機伺服器上的總交換檔案大小，以位元組為單位。

UsedBytes

主機伺服器上使用的交換檔案大小，以位元組為單位。

Aws

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

Region

Amazon EC2 執行個體 AWS 的區域。

Ec2InstanceType

Amazon EC2 執行個體的類型。

Ec2InstanceId

Amazon EC2 執行個體的 ID。

Ec2ImageId

Amazon EC2 映像的 ID。

CpuInfo

主機伺服器的 CPUs的相關資訊。此資料結構包含下列巢狀回應參數：

Vendor

主機伺服器的 CPU 廠商。

ModelName

主機伺服器的 CPU 模型名稱。

Architecture

主機伺服器的 CPU 架構。

NumberOfCpus

主機伺服器上CPUs 數量。

PhysicalCorePerCpu

每個 CPU 的 CPU 核心數量。

CpuLoadAverage

主機伺服器的 CPU 負載相關資訊。此資料結構包含下列巢狀回應參數：

OneMinute

過去 1 分鐘期間的平均 CPU 負載。

FiveMinutes

過去 5 分鐘期間的平均 CPU 負載。

FifteenMinutes

過去 15 分鐘期間的平均 CPU 負載。

Gpus

主機伺服器的 GPUs 相關資訊。此資料結構包含下列巢狀回應參數：

Vendor

主機伺服器的 GPU 廠商。

ModelName

主機伺服器的 GPU 模型名稱。

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

已登入使用者的使用者名稱。

Tags

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

下列範例說明所有可用的 Amazon DCV 伺服器。系統會分頁結果，以顯示每個頁面的兩個結果。

```
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

以下是範例輸出。


```
{
    "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

描述一或多個 Amazon DCV 工作階段。

主題

請求參數

`SessionIds`

要描述之工作階段IDs。

類型：字串

必要：否

`NextToken`

用來擷取結果下一頁的字符。

類型：字串

必要：否

`Filters`

要套用至請求的其他篩選條件。支援的篩選條件包括：

- tag：key - 指派給工作階段的標籤。
- owner - 工作階段擁有者。

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

工作階段執行所在伺服器的相關資訊。此資料結構包含下列巢狀回應參數：

Ip

Amazon DCV 伺服器主機的 IP 地址。

Hostname

Amazon DCV 伺服器主機的主機名稱。

Port

Amazon DCV 伺服器與 Amazon DCV 用戶端通訊的連接埠。

Endpoints

Amazon DCV 伺服器端點的相關資訊。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

伺服器端點所使用的通訊協定。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

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

工作階段的目前狀態。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。
- 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。

Substate

工作階段的目前子狀態。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

工作階段建立的日期和時間。

LastDisconnectionTime

上次用戶端中斷連線的日期和時間。

NumOfConnections

作用中用戶端連線的數量。

StorageRoot

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

類型：字串

必要：否

範例

Python

請求

下列範例說明由 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

以下是範例輸出。

```
{
    "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 Amazon 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

要刪除的工作階段 ID。

類型：字串

必要：是

Owner

要刪除之工作階段的擁有者。

類型：字串

必要：是

Force

從代理程式的快取中移除工作階段，並嘗試從 Amazon DCV 伺服器刪除該工作階段。這有助於從代理程式快取中移除過時的工作階段。例如，如果 Amazon DCV 伺服器已停止，但工作階段仍在代理程式上註冊，請使用此旗標清除代理程式快取中的工作階段。

請記住，如果工作階段仍然作用中，則代理程式會重新快取。

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

只有在工作階段已成功刪除時才會傳回。指示工作階段的目前狀態。如果請求成功完成，工作階段會轉換為 DELETING 狀態。刪除工作階段可能需要幾分鐘的時間。刪除後，狀態會從 轉換為 DELETING DELETED。

FailureReason

只有在無法刪除某些工作階段時才會傳回。指出無法刪除工作階段的原因。

範例

Python

請求

下列範例會刪除兩個工作階段：一個 ID 為 SessionId123的工作階段，user1以及一個 ID 為 SessionIdabc的工作階段user99。

```
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

以下是範例輸出。 SessionId123 已成功刪除，而 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

取得特定使用者與特定 Amazon DCV 工作階段連線的連線資訊。

主題

請求參數

`SessionId`

要檢視連線資訊的工作階段 ID。

類型：字串

必要：是

`User`

要檢視連線資訊的使用者名稱。

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

工作階段執行所在伺服器的相關資訊。此資料結構包含下列巢狀回應參數：

Ip

Amazon DCV 伺服器主機的 IP 地址。

Hostname

Amazon DCV 伺服器主機的主機名稱。

Port

Amazon DCV 伺服器與 Amazon DCV 用戶端通訊的連接埠。

Endpoints

Amazon DCV 伺服器端點的相關資訊。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

伺服器端點所使用的通訊協定。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

Amazon DCV 伺服器組態檔案的路徑。

Tags

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

工作階段的目前狀態。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。
- 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。

CreationTime

工作階段建立的日期和時間。

LastDisconnectionTime

上次用戶端中斷連線的日期和時間。

NumOfConnections

使用者與工作階段的並行連線數。

ConnectionToken

用於連線至工作階段的身分驗證字符。

其他資訊

從此 API 取得的資訊可以傳遞至 Amazon DCV 用戶端，以連線至 Amazon DCV 工作階段。

如果是 Amazon DCV Web 用戶端，您可以建置可在瀏覽器中開啟的 URL。URL 的格式如下：

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

如果是 Amazon DCV 原生用戶端，您可以使用dcv://結構描述建置 URL。安裝 Amazon DCV 原生用戶端時，它會向系統註冊本身，做為 dcv:// URLs的處理常式。URL 的格式如下：

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

> **Note**
>
> 如果您使用的是 Amazon EC2，IP 地址應為公有地址。如果您的組態在閘道後方有 Amazon DCV 主機，請指定閘道地址，而不是 SessionConnectionData API 傳回的地址。

範例

Python

請求

下列範例會取得使用者名稱為 的使用者連線資訊，user1以及 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

以下是範例輸出。

```
{
    "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

取得一或多個 Amazon DCV 工作階段的螢幕擷取畫面。

若要修改映像格式，請在 Session Manager Broker 組態上設定 `session-screenshot-format` 參數。請參閱《Amazon DCV Session Manager 管理員指南》中的代理程式組態檔案。

未指定`GetSessionScreenshots`請求的 `MaxWidth`或 `MaxHeight` 參數時，將使用 Session Manager Broker 組態檔案中設定的 `session-screenshot-max-width`和 `session-screenshot-max-height`值。若要修改這些參數，另請參閱《Amazon DCV Session Manager 管理員指南》中的代理程式組態檔案。

螢幕擷取畫面解析度的上限值僅限於遠端工作階段解析度。如果 `MaxWidth`和 `MaxHeight` 參數設定為高於目前遠端工作階段解析的值，產生的螢幕擷取畫面將僅限於實際工作階段解析。

> **Note**
>
> 若要從存取主控台修改這些值，請參閱《Amazon DCV 存取主控台管理員指南》中的 Web 用戶端組態檔案。若要使用 Session Manager CLI 修改這些值，請參閱《Amazon DCV CLI 指南`get-session-screenshots`》中的 。

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 Amazon DCV 工作階段 ID。

類型：字串

必要：是

MaxWidth

工作階段螢幕擷取畫面的最大寬度，以像素為單位。如果未指定，將套用來自 Session Manager Broker 組態的值。如果提供，這必須是大於 0 的數字。

類型：整數

必要：否

MaxHeight

工作階段螢幕擷取畫面的最大高度，以像素為單位。如果未指定，將套用來自 Session Manager Broker 組態的值。如果提供，這必須是大於 0 的數字。

類型：整數

必要：是

回應參數

RequestId

請求的唯一 ID。

SuccessfulList

成功螢幕擷取畫面的相關資訊。此資料結構包含下列巢狀回應參數：

SessionScreenshot

螢幕擷取畫面的相關資訊。此資料結構包含下列巢狀回應參數：

SessionId

擷取螢幕擷取畫面的 Amazon DCV 工作階段 ID。

Images

映像的相關資訊。此資料結構包含下列巢狀回應參數：

Format

映像的格式。可能的值包括：jpeg 和 png。

Data

螢幕擷取畫面影像 base64 編碼格式。

CreationTime

擷取螢幕擷取畫面的日期和時間。

Primary

指出螢幕擷取畫面是否為 Amazon DCV 工作階段的主要顯示畫面。

UnsuccessfulList

有關失敗螢幕擷取畫面的資訊。此資料結構包含下列巢狀回應參數：

GetSesionScreenshotRequestData

失敗的原始請求。

SessionId

擷取螢幕擷取畫面的 Amazon DCV 工作階段 ID。

FailureReason

失敗的原因。

GetSessionScreenshotRequestData

失敗的原始請求。

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，其最大寬度設定為 800，最大高度設定為 600。工作階段sessionId2不存在，並導致失敗。

```
from swagger_client.models.describe_servers_request_data import
 GetSessionScreenshotRequestData

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, max_width=None, max_height=None):
    request = [GetSessionScreenshotRequestData(session_id=session_id,
 max_width=max_width, max_height=max_height) 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"]), 800, 600)
```

回應

以下是範例輸出。

```
{
    "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 Amazon DCV 伺服器。在 Amazon DCV 伺服器上建立 Amazon DCV 工作階段之前，您必須變更伺服器的狀態以開啟。Amazon DCV 伺服器開啟後，您可以在伺服器上建立 Amazon DCV 工作階段。

主題

請求參數

`ServerId`

要開啟的伺服器 ID。

類型：字串

必要：是

回應參數

`RequestId`

請求的唯一 ID。

`SuccessfulList`

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

`ServerId`

已成功開啟的伺服器 ID。

`UnsuccessfulList`

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

`OpenServerRequestData`

失敗的原始請求的相關資訊。此資料結構包含下列巢狀回應參數：

ServerId

無法開啟的 Amazon DCV 伺服器 ID。

FailureCode

失敗的程式碼。

FailureReason

失敗的原因。

範例

Python

請求

下列範例會開啟兩個 Amazon 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

以下是範例輸出。

```
{
    "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 Amazon DCV 工作階段的使用者許可。

主題

請求參數

`SessionId`

要變更許可的工作階段 ID。

類型：字串

必要：是

`Owner`

要變更其許可的工作階段擁有者。

類型：字串

必要：是

PermissionFile

要使用之許可檔案的 Base64-encoded內容。如需詳細資訊，請參閱《Amazon DCV 管理員指南》中的設定 Amazon 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 的工作階段設定新的許可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

以下是範例輸出。

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

Amazon DCV Session Manager 的版本備註和文件歷史記錄

此頁面提供 Amazon DCV Session Manager 的版本備註和文件歷史記錄。

主題

Amazon DCV Session Manager 版本備註

本節提供 Amazon DCV Session Manager 主要更新、功能版本和錯誤修正的概觀。所有更新都會依發行日期進行組織。我們會經常更新文件，以處理您傳送給我們的意見回饋。

主題

- 2021.1-10557 — 2021 年 5 月 31 日
- 2021.0-10242 — 2021 年 4 月 12 日
- 2020.2-9662 — 2020 年 12 月 4 日
- 2020.2-9508 — 2020 年 11 月 11 日

2024.0-531 — 2025 年 6 月 17 日

建置編號	變更與錯誤修正
• 中介裝置：531 • 代理程式：852 • CLI：154	• 新增在過期前續約憑證的功能。 • 將 NICE DCV 重新命名為 Amazon DCV。 • 錯誤修正。

2024.0-504 — 2025 年 3 月 31 日

建置編號	變更與錯誤修正
• 中介裝置：504 • 代理程式：817 • CLI：154	• 新增對 AL2023 的支援。 • 錯誤修正與效能改進。

2024.0-493 — 2025 年 1 月 15 日

建置編號	變更與錯誤修正
• 中介裝置：493 • 代理程式：801 • CLI：152	• 新增參數至`GetSessionScreenshot` 請求，以指定螢幕擷取畫面的最大高度和寬度。 • 新增參數至中介裝置組態檔案，指定從系統刪除無法連線 Amazon DCV 伺服器上工作階段的秒數。 • 修正未遵守中介裝置組態檔案中的 `seconds-before-deleting-unreachable-dcv-server` 參數的問題。 • 錯誤修正與效能改進。

2024.0-457 — 2024 年 10 月 1 日

建置編號	變更與錯誤修正
• 中介裝置：457 • 代理程式：748 • CLI：140	• 將 NICE DCV 重新命名為 Amazon DCV。 • 新增對 Ubuntu 24.04 的支援。

2023.1-17652 — 2024 年 8 月 1 日

建置編號	變更與錯誤修正
• 中介裝置：426 • 代理程式：748 • CLI：140	• 錯誤修正與效能改進。

2023.1-16388 — 2024 年 6 月 26 日

建置編號	變更與錯誤修正
• 中介裝置：417 • 代理程式：748 • CLI：140	• 已修正錯誤顯示記憶體為 TB 而非 GB 的錯誤。 • 錯誤修正與效能改進。

2023.1 — 2023 年 11 月 9 日

建置編號	變更與錯誤修正
• 中介裝置：410 • 代理程式：732 • CLI：140	• 錯誤修正與效能改進

2023.0-15065 — 2023 年 5 月 4 日

建置編號	變更與錯誤修正
• 中介裝置：392 • 代理程式：675 • CLI：132	• 新增對 ARM 平台上 Red Hat Enterprise Linux 9、Rocky Linux 9 和 CentOS Stream 9 的支援。

2023.0-14852 — 2023 年 3 月 28 日

建置編號	變更與錯誤修正
• 中介裝置：392 • 代理程式：642 • CLI：132	• 新增對 Red Hat Enterprise Linux 9、Rocky Linux 9 和 CentOS Stream 9 的支援。

2022.2-13907 — 2022 年 11 月 11 日

建置編號	變更與錯誤修正
• 中介裝置：382 • 代理程式：612 • CLI：123	• 在`DescribeSessions` 回應中新增`Substate`欄位。 • 已修正根據使用中的 URL，可能導致 CLI 無法連線至代理程式的問題。

2022.1-13067 — 2022 年 6 月 29 日

建置編號	變更與錯誤修正
• 中介裝置：355 • 代理程式：592 • CLI：114	• 新增在 Graviton AWS 執行個體上執行代理程式的支援。 • 新增了對 Ubuntu 22.04 的代理程式和代理程式支援。

2022.0-11952 — 2022 年 2 月 23 日

建置編號	變更與錯誤修正
• 中介裝置：341 • 代理程式：520 • CLI：112	• 已將日誌輪換功能新增至 代理程式。 • 新增在中介裝置中設定 Java 首頁的組態參數。 • 改善從快取到中介裝置中磁碟的資料清除。 • 已修正 CLI 中的 URL 驗證。

2021.3-11591 — 2021 年 12 月 20 日

建置編號	新功能
• 中介裝置：307 • 代理程式：453 • CLI：92	• 新增了與 Amazon DCV Connection Gateway 整合的支援。 • 新增了對 Ubuntu 18.04 和 Ubuntu 20.04 的中介裝置支援。

2021.2-11445 — 2021 年 11 月 18 日

建置編號	變更與錯誤修正
• 中介裝置：288 • 代理程式：413 • CLI：54	• 修正包含 Windows 網域的登入名稱驗證問題。

2021.2-11190 — 2021 年 10 月 11 日

建置編號	變更與錯誤修正
• 中介裝置：254 • 代理程式：413 • CLI：54	• 修正命令列界面中無法啟動 Windows 工作階段的問題。

2021.2-11042 — 2021 年 9 月 1 日

建置編號	新功能	變更與錯誤修正
• 中介裝置：254 • 代理程式：413 • CLI：37	• Amazon DCV Session Manager 現在提供命令列界面 (CLI) 支援。您可以在 CLI 中建立和管理 Amazon DCV 工作階段，而不是呼叫 APIs。 • Amazon DCV Session Manager 推出中介裝置資料持久性。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 Web 權杖的演算法。透過此變更，您可以使用 Azure AD 做為外部授權伺服器。

2021.1-10557 — 2021 年 5 月 31 日

建置編號	新功能	變更與錯誤修正
• 中介裝置：214 • 代理程式：365	• Amazon DCV Session Manager 新增了對傳入 Linux 上自動執行檔案的輸入參數的支援。 • 伺服器屬性現在可以做為需求傳遞至 CreateSessions API。	• 我們修正了 Windows 上自動執行檔案的問題。

2021.0-10242 — 2021 年 4 月 12 日

建置編號	變更與錯誤修正

• 中介裝置：183
• 代理程式：318
	• Amazon DCV Session Manager 推出下列新 APIs：
	  • OpenServers
	  • CloseServers
	  • DescribeServers
	  • GetSessionScreenshots

建置編號	變更與錯誤修正

	• 它還引入了以下新組態參數：
	  • 中介裝置參數：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 — 2020 年 12 月 4 日

建置編號	變更與錯誤修正
• 中介裝置：114 • 代理程式：211	• 我們修正了自動產生的 TLS 憑證導致代理程式無法啟動的問題。

2020.2-9508 — 2020 年 11 月 11 日

建置編號	變更與錯誤修正
• 中介裝置：78 • 代理程式：183	• Amazon DCV Session Manager 的初始版本。

文件歷史紀錄

下表說明此版本 Amazon DCV Session Manager 的文件。

變更	描述	日期
Amazon DCV 2024.0-531 版	Amazon DCV Session Manager 已更新為 Amazon DCV 2024.0-531。如需詳細資訊，請參閱???。	2025 年 6 月 17 日
Amazon DCV 2024.0-504 版	Amazon DCV Session Manager 已更新為 Amazon DCV 2024.0-504。如需詳細資訊，請參閱???。	2025 年 3 月 31 日
Amazon DCV 2024.0-493 版	Amazon DCV Session Manager 已更新至 Amazon DCV 2024.0-493。如需詳細資訊，請參閱2024.0-493 — 2025 年 1 月 15 日。	2025 年 1 月 15 日
Amazon DCV 2024.0-457 版	Amazon DCV Session Manager 已更新至 Amazon DCV 2024.0-457。如需詳細資訊，請參閱2024.0-457 — 2024 年 10 月 1 日。	2024 年 9 月 30 日
Amazon DCV 2023.1-17652 版	Amazon DCV Session Manager 已更新 Amazon DCV 2023.1-17652。如需詳細資訊，請參閱2023.1-17652 — 2024 年 8 月 1 日。	2024 年 8 月 1 日
Amazon DCV 2023.1-16388 版	Amazon DCV Session Manager 已更新為 Amazon DCV 2023.1-16388。如需詳細資訊，請參閱2023.1-16388 — 2024 年 6 月 26 日。	2024 年 6 月 26 日

變更	描述	日期
Amazon DCV 2023.1 版	Amazon DCV Session Manager 已更新至 Amazon DCV 2023.1。如需詳細資訊，請參閱2023.1 — 2023 年 11 月 9 日。	2023 年 11 月 9 日
Amazon DCV 2023.0 版	Amazon DCV Session Manager 已更新至 Amazon DCV 2023.0。如需詳細資訊，請參閱2023.0-14852 — 2023 年 3 月 28 日。	2023 年 3 月 28 日
Amazon DCV 2022.2 版	Amazon DCV Session Manager 已更新至 Amazon DCV 2022.2。如需詳細資訊，請參閱2022.2-13907 — 2022 年 11 月 11 日。	2022 年 11 月 11 日
Amazon DCV 2022.1 版	Amazon DCV Session Manager 已更新至 Amazon DCV 2022.1。如需詳細資訊，請參閱2022.1-13067 — 2022 年 6 月 29 日。	2022 年 6 月 29 日
Amazon DCV 2022.0 版	Amazon DCV Session Manager 已更新至 Amazon DCV 2022.0。如需詳細資訊，請參閱2022.0-11952 — 2022 年 2 月 23 日。	2022 年 2 月 23 日
Amazon DCV 2021.3 版	Amazon DCV Session Manager 已更新至 Amazon DCV 2021.3。如需詳細資訊，請參閱2021.3-11591 — 2021 年 12 月 20 日。	2021 年 12 月 20 日
Amazon DCV 2021.2 版	Amazon DCV Session Manager 已更新至 Amazon DCV 2021.2。如需詳細資訊，請參閱2021.2-11042 — 2021 年 9 月 1 日。	2021 年 9 月 1 日

變更	描述	日期
Amazon DCV 2021.1 版	Amazon DCV Session Manager 已更新至 Amazon DCV 2021.1。如需詳細資訊，請參閱2021.1-10557 — 2021 年 5 月 31 日。	2021 年 5 月 31 日
Amazon DCV 2021.0 版	Amazon DCV Session Manager 已更新至 Amazon DCV 2021.0。如需詳細資訊，請參閱2021.0-10242 — 2021 年 4 月 12 日。	2021 年 4 月 12 日
Amazon DCV Session Manager 的初始版本	此內容的第一個發佈。	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。