デベロッパーガイド

Amazon DCV セッションマネージャー

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon DCV セッションマネージャー: デベロッパーガイド

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは、Amazon のものではない製品またはサービスにも関連して、お客様に混乱を招いたり Amazon の信用を傷つけたり失わせたりするいかなる形においても使用することはできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

セッションマネージャーとは 1
セッションマネージャーの仕組み 1
機能 3
セッションマネージャー API の開始方法 5
ステップ 1: API クライアントを生成する 5
ステップ 2: クライアント API を登録する 6
ステップ 3: アクセストークンを取得して API リクエストを行う 7
セッションマネージャー API リファレンス 10
CloseServers 10
リクエストパラメータ 7
レスポンスパラメータ 11
例 12
CreateSessions 13
リクエストパラメータ 7
レスポンスパラメータ 11
例 12
DescribeServers 21
リクエストパラメータ 7
レスポンスパラメータ 11
例 12
DescribeSessions 32
リクエストパラメータ 7
レスポンスパラメータ 11
例 12
DeleteSessions 39
リクエストパラメータ 7
レスポンスパラメータ 11
例 12
GetSessionConnectionData 41
リクエストパラメータ 7
レスポンスパラメータ 11
追加情報 45
例 12
GetSessionScreenshots 47

リクエストパラメータ 7
レスポンスパラメータ 11
例 12
OpenServers 52
リクエストパラメータ 7
レスポンスパラメータ 11
例 12
UpdateSessionPermissions 54
リクエストパラメータ 7
レスポンスパラメータ 11
例 12
リリースノートとドキュメント履歴 57
リリースノート 57
2024.0-531 — 2025 年 6 月 17 日 58
2024.0-504 — 2025 年 3 月 31 日 58
2024.0-493 — 2025 年 1 月 15 日 58
2024.0-457 - 2024 年 10 月 1 日 59
2023.1-17652 - 2024 年 8 月 1 日 59
2023.1-16388 - 2024 年 6 月 26 日 59
2023.1— 2023 年 11 月 9 日 60
2023.0-15065— 2023 年 5 月 4 日 60
2023.0-14852— 2023 年 3 月 28 日 60
2022.2-13907— 2022 年 11 月 11 日 60
2022.1-13067— 2022 年 6 月 29 日 61
2022.0-11952 — 2022 年 2 月 23 日 61
2021.3-11591 — 2021 年 12 月 20 日 61
2021.2-11445 — 2021 年 11 月 18 日 62
2021.2-11190 — 2021 年 10 月 11 日 62
2021.2-11042 — 2021 年 9 月 1 日 62
2021.1-10557 — 2021 年 5 月 31 日 63
2021.0-10242 — 2021 年 4 月 12 日 63
2020.2-9662 — 2020 年 12 月 4 日 64
..... 64
ドキュメント履歴 64
..... lxviii

Amazon DCV セッションマネージャーとは

> **Note**
>
> Amazon DCV は以前は NICE DCV と呼ばれていました。

Amazon DCV セッションマネージャーとは、インストール可能なソフトウェアパッケージ (エージェントとブローカー) とアプリケーションプログラミングインターフェイス (API) のセットです。デベロッパーや独立系ソフトウェアベンダー (ISV) がこれを使えば、Amazon DCV サーバーフリートにおける Amazon DCV セッションのライフサイクルの作成および管理をプログラムで実行できるフロントエンドアプリケーションを簡単に構築できます。

このガイドでは、セッションマネージャー API を使用して Amazon DCV セッションのライフサイクルを管理する方法について説明します。セッションマネージャーブローカーとエージェントをインストールして設定する方法の詳細については、「Amazon DCV セッションマネージャー管理者ガイド」を参照してください。

前提条件

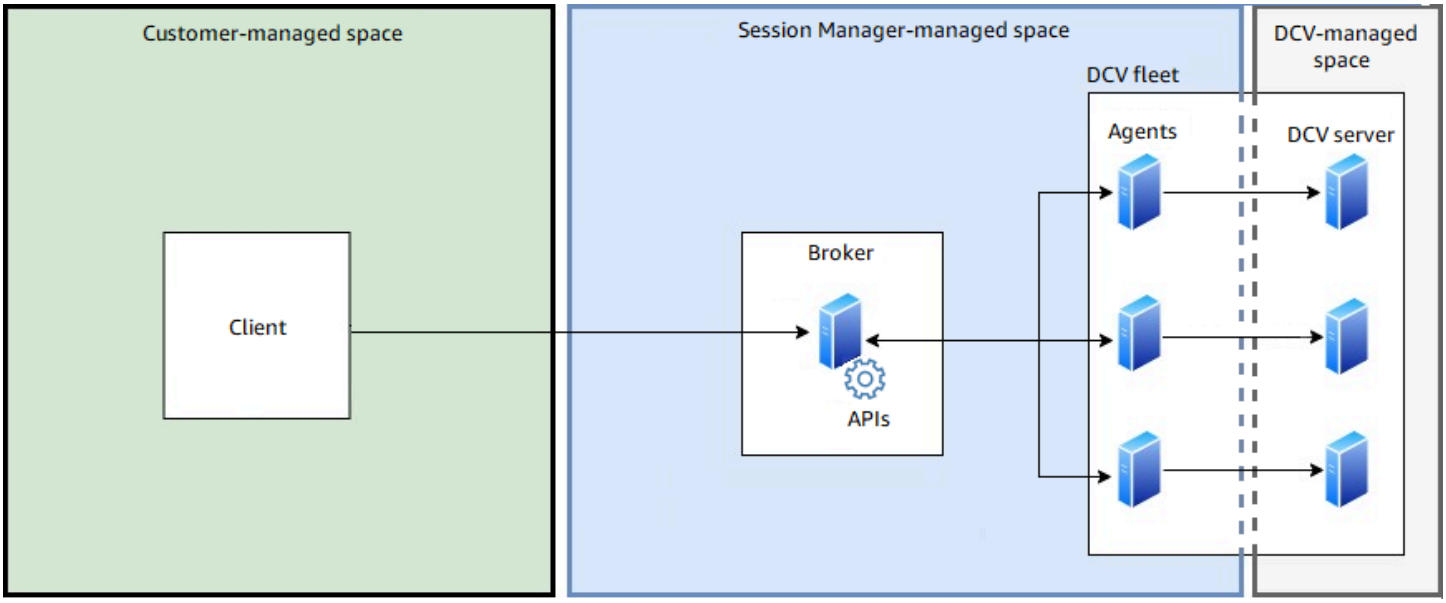
セッションマネージャー API の使用を開始する前に、Amazon DCV と Amazon DCV セッションについて理解しておいてください。詳細については、「Amazon DCV 管理者ガイド」を参照してください。

トピック

- セッションマネージャーの仕組み
- 機能

セッションマネージャーの仕組み

次の図は、セッションマネージャーの高レベルコンポーネントを示しています。



ブローカー

ブローカーは、セッションマネージャー API のホストと公開を実行するウェブサーバーです。API リクエストを受信して処理し、クライアントから Amazon DCV セッションを管理し、関連するエージェントに指示を渡します。ブローカーは、Amazon DCV サーバーとは別のホストにインストールする必要がありますが、クライアントへのアクセスが可能な状態で、かつ、エージェントにアクセスできる必要があります。

Agent

エージェントは、フリート内の各 Amazon DCV サーバーにインストールされます。エージェントは、ブローカーからの指示を受信し、それぞれの Amazon DCV サーバーでそれらを実行します。さらに、Amazon DCV サーバーの状態を監視し、定期的にステータス更新をブローカーに送信します。

API

セッションマネージャーでは、Amazon DCV サーバーのフリートで Amazon DCV セッションを管理するために使用できる REST アプリケーションプログラミングインターフェイス (API) のセットが公開されます。API はブローカーでホストされて公開されます。デベロッパーは API を呼び出せるカスタムのセッション管理クライアントを構築できます。

クライアント

クライアントは、ブローカーにより公開されるセッションマネージャー API を呼び出すために開発されるフロントエンドのアプリケーションまたはポータルです。エンドユーザーは、クライ

アントを使用して、フリート内の Amazon DCV サーバーでホストされるセッションを管理します。

アクセストークン

API リクエストを実行するには、アクセストークンを提供する必要があります。トークンは、登録されたクライアント API によって、ブローカーまたは外部認可サーバーから要求できます。トークンをリクエストしてアクセスするには、クライアント API から有効な認証情報を提供される必要があります。

クライアント API

クライアント API は、Swagger Codegen を使用して、セッションマネージャー API 定義 YAML ファイルから生成されます。クライアント API は API リクエストの作成に使用されます。

Amazon DCV セッション

Amazon DCV セッションは、Amazon DCV サーバーがクライアントからの接続を受け入れることができる期間です。クライアントが Amazon DCV セッションに接続できるようにするには、Amazon DCV サーバーで Amazon DCV セッションを作成する必要があります。Amazon DCV はコンソールセッションと仮想セッションの両方をサポートし、各セッションには指定された所有者と一連のアクセス許可があります。セッションマネージャー API を使用して、Amazon DCV セッションのライフサイクルを管理します。Amazon DCV セッションは、次のいずれかの状態になります。

- `CREATING` — ブローカーはセッション作成中です。
- `READY` — セッションはクライアント接続を受け入れる準備ができました。
- `DELETING` — セッションが削除されています。
- `DELETED` — セッションが削除されました。
- `UNKNOWN`— セッションの状態を判別できません。ブローカーとエージェントが通信できない可能性があります。

機能

DCV セッションマネージャーには以下の特徴があります。

- Amazon DCV セッション情報を提供 — 複数の Amazon DCV サーバーで実行されているセッションに関する情報が得られます。

- 複数の Amazon DCV セッションのライフサイクルを管理 — 1 件の API リクエストにより、複数の Amazon DCV サーバー間で複数のユーザーに対して複数のセッションの作成や削除を行うことができます。
- タグをサポート — カスタムタグを使用してセッション作成時に Amazon DCV サーバーのグループをターゲットにすることができます。
- 複数の Amazon DCV セッションの許可を管理 — 1 件の API リクエストで複数のセッションのユーザー許可に変更を加えることができます。
- 接続情報を提供 — Amazon DCV セッションのクライアント接続情報が得られます。
- クラウドとオンプレミスのサポート — AWS、オンプレミス、または代替のクラウドベースサーバーでセッションマネージャーを使用できます。

セッションマネージャー API の開始方法

Amazon DCV セッションマネージャー API は、リモートデスクトップセッションを管理するための自動インターフェイスを提供します。この API を使用して、デベロッパーは DCV セッションをプログラムで作成、一覧表示、開始、停止、制御できます。これにより、Amazon DCV 機能をカスタムアプリケーションやワークフローに統合できます。この API を活用することで、組織はリモート視覚化ワークロードの管理を合理化し、一般的なタスクの多くを自動化できます。

Amazon DCV API への呼び出しを開始する前に、アプリケーションを認証し、必要なリソースへのアクセスを許可するアクセストークンを取得する必要があります。Amazon DCV API は認証に OAuth 2.0 を使用するため、アプリケーションを登録して必要な認証情報を取得する必要があります。アクセストークンを取得したら、Amazon DCV API エンドポイントへのリクエストの送信を開始して、データの処理を開始できます。

トピック

- ステップ 1: API クライアントを生成する
- ステップ 2: クライアント API を登録する
- ステップ 3: アクセストークンを取得して API リクエストを行う

ステップ 1: API クライアントを生成する

セッションマネージャー API は、単一の YAML ファイル内で定義されます。API は OpenAPI3.0 仕様に基づいています。この仕様では、RESTful API に対して、標準的で言語に依存しないインターフェースが定義されています。詳細については、「OpenAPI の仕様」を参照してください。

YAML ファイルを使用すると、サポートされている言語のうちの 1 つで API クライアントを生成できます。これを実行するには、Swagger Codegen 3.0 以降を使用する必要があります。サポートされている言語の詳細については、「swagger-codegen repo」を参照してください。

API クライアントを生成する方法

1. セッションマネージャーブローカーからセッションマネージャー API YAML ファイルをダウンロードします。YAML ファイルは次の URL から入手できます。

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

2. Swagger Codegen をインストールします。

- macOS

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

- その他のプラットフォーム

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

```
$ cd swagger-codegen
```

3. API クライアントを生成します。

- macOS

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

- その他のプラットフォーム

```
$ 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
```

ステップ 2: クライアント API を登録する

API リクエストは、アクセストークンを使用して認証情報を検証します。これらの認証情報は、クライアントがブローカーに登録されたときに生成されるクライアント ID とクライアントパスワードに基づきます。

このトークンにアクセスするには、ブローカーに登録する必要があります。register-api-client を使用してクライアント API を登録します。

お使いのクライアントのクライアント ID とクライアントパスワードをお持ちでない場合は、ブローカー管理者からそれらをリクエストする必要があります。

ステップ 3: アクセストークンを取得して API リクエストを行う

この例では、アクセストークンをセットアップするステップを順を追って説明し、基本的な API リクエストを行う方法を示します。これにより、Amazon DCV API を搭載したより高度なアプリケーションの構築を開始するための基本的な知識が得られます。

この例では、この操作方法を `DescribeSessions` API を使用して説明します。

Example

まず、アプリケーションに必要なモデルをインポートします。

クライアント ID (`__CLIENT_ID`)、クライアントパスワード (`__CLIENT_SECRET`)、および、ポート番号を含むブローカー URL (`__PROTOCOL_HOST_PORT`) の変数を宣言します。

次に、クライアント認証を生成する `build_client_credentials` という関数を作成します。クライアント認証情報を生成するには、まずクライアント ID とクライアントパスワードを連結し、値をコロン (*`client_ID`*`:`*`client_password`*) で区切って、文字列全体に対して Base64 エンコードを行う必要があります。

```
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')
```

これでクライアント認証情報を取得したので、これを使用してブローカーからアクセストークンをリクエストできます。これを実行するには、get_access_token という関数を作成します。POST で https://*Broker_IP*:*8443*/oauth2/token?grant_type=client_credentials を呼び出し、基本エンコードクライアント認証情報と application/x-www-form-urlencoded のコンテンツタイプを含む認可ヘッダーを提供します。

```
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
```

ここで、クライアント API のインスタンスの作成に必要な関数を作成します。クライアント API のインスタンスを作成するには、クライアント設定と、リクエストに使用するヘッダーを指定する必要があります。get_client_configuration 関数により設定オブジェクトが作成されます。このオブジェクトには、ブローカーの IP アドレスとポート、および、ブローカーの管理者から受け取ったブローカーの自己署名証明書へのパスが含まれます。set_request_headers 関数により、クライアント認証情報とアクセストークンが含まれているリクエストヘッダーオブジェクトが作成されます。

```
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
```

最後に、DescribeSessions API を呼び出すメインメソッドを作成します。詳細については、「DescribeSessions」を参照してください。

```
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'}])
```

セッションマネージャー API リファレンス

このリファレンスでは、独自のシステムでセッションマネージャー API を効果的に活用できるように、使用可能な API アクション、必要なパラメータ、およびレスポンス形式の詳細を提供します。セッションマネージャー API を使用すると、インタラクティブセッションを開始、停止、および詳細を取得できます。これにより、機能を自動化してアプリケーションやワークフローに統合できます。

トピック

- CloseServers
- CreateSessions
- DescribeServers
- DescribeSessions
- DeleteSessions
- GetSessionConnectionData
- GetSessionScreenshots
- OpenServers
- UpdateSessionPermissions

CloseServers

1 つまたは複数の Amazon DCV サーバーを閉じます。Amazon DCV サーバーを閉じると、Amazon DCV セッション配置で使用できなくなります。閉じたサーバーでは Amazon DCV セッションを作成することはできません。サーバーを閉じると、サーバーでセッションが実行されなくなり、ユーザーがサーバーで新しいセッションを作成できなくなります。

トピック

- リクエストパラメータ
- レスポンスパラメータ
- 例

リクエストパラメータ

ServerId

閉じるサーバーの ID。

タイプ: 文字列

必須: はい

Force

終了操作を強制します。`true` を指定した場合、実行中のセッションがあってもサーバーが閉じられます。セッションは引き続き実行されます。

型: ブール値

必須: いいえ

レスポンスパラメータ

RequestId

リクエストの一意の ID。

SuccessfulList

正常に閉じられた Amazon DCV サーバーに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

ServerId

正常に閉じられたサーバーの ID。

UnsuccessfulList

閉じられなかった Amazon DCV サーバーに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

CloseServerRequestData

失敗した元のリクエストに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

ServerId

閉じられなかった Amazon DCV サーバーの ID。

Force

リクエストされた強制パラメータ。

FailureCode

失敗のコード。

FailureReason

失敗の理由。

例

Python

リクエスト

次の例では、2 つの Amazon DCV サーバー (`serverId1` と `serverId2`) を閉じます。サーバー `serverId2` は存在せず、失敗します。

```
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"])
```

レスポンス

以下は出力例です。

```
{
    "RequestId": "4d7839b2-a03c-4b34-a40d-06c8b21099e6",
    "SuccessfulList": [
        {
            "ServerId": "serverId1"
        }
    ],
    "UnsuccessfulList": [
        {
            "OpenServerRequestData": {
                "ServerId": "serverId2"
            },
            "FailureCode": "DCV_SERVER_NOT_FOUND",
            "FailureReason": "Dcv server not found."
        }
    ]
}
```

CreateSessions

指定された詳細で新しい Amazon DCV セッションを作成します。

API アクション

- リクエストパラメータ
- レスポンスパラメータ
- 例

リクエストパラメータ

Name

セッションの名前。

タイプ: 文字列

必須: はい

Owner

セッション所有者の名前。これは、ターゲットの Amazon DCV サーバーの既存ユーザーの名前でなければなりません。

タイプ: 文字列

必須: はい

Type

セッションのタイプ。セッションタイプの詳細については、「Amazon DCV 管理者ガイド」の「Amazon DCV セッションの概要」を参照してください。

有効な値: CONSOLE | VIRTUAL

タイプ: 文字列

必須: はい

InitFile

Linux Amazon DCV サーバーの仮想セッションでサポートされています。Windows および Linux Amazon DCV サーバーのコンソールセッションではサポートされません。セッションを作成時に初期化するために実行する Amazon DCV サーバー上のカスタムスクリプトへのパス。このファイルパスは、`agent.init_folder` エージェント設定パラメータに対して指定された init ディレクトリに関連します。ファイルが指定された init ディレクトリ内にある場合は、ファイル名のみを指定します。ファイルが指定された init ディレクトリ内にない場合は、相対パスを指定します。詳細については、「Amazon DCV セッションマネージャー管理者ガイド」の「エージェント設定ファイル」を参照してください。

タイプ: 文字列

必須: いいえ

MaxConcurrents

同時 Amazon DCV クライアントの最大数。

タイプ: 整数

必須: いいえ

`DcvGlEnabled`

仮想セッションがハードウェアベースの OpenGL を使用するように設定されているかどうかを示します。仮想セッションでのみサポートされます。このパラメータは Windows Amazon DCV サーバーではサポートされません。

有効な値: true | false

型: ブール値

必須: いいえ

`PermissionsFile`

アクセス許可ファイルの Base64 エンコードコンテンツ。省略した場合、デフォルトはサーバーのデフォルトです。詳細については、「Amazon DCV 管理者ガイド」の「Amazon DCV 認可の設定」を参照してください。

タイプ: 文字列

必須: いいえ

`EnqueueRequest`

すぐに受理できない場合に、リクエストをキューに入れるかどうかを示します。

型: ブール値

デフォルト: false

必須: いいえ

`AutorunFile`

Windows Amazon DCV サーバーのコンソールセッションおよび Linux Amazon DCV サーバーの仮想セッションでサポートされています。Linux Amazon DCV サーバーのコンソールセッションではサポートされません。

セッション内で実行されるホストサーバー上のファイルへのパス。このファイルパスは、`agent.autorun_folder` エージェント設定パラメータに対して指定された自動実行ディレクトリに関連します。ファイルが指定された autorun ディレクトリ内にある場合は、ファイル名のみを指定します。ファイルが指定された autorun ディレクトリにない場合は、相対パスを指定します。詳細については、「Amazon DCV セッションマネージャー管理者ガイド」の「エージェント設定ファイル」を参照してください

ファイルは、指定された所有者に代わって実行されます。指定された所有者は、サーバーでファイルを実行するためのアクセス許可を持っている必要があります。Windows Amazon DCV サーバーでは、所有者がセッションにログインしたときにファイルが実行されます。Linux Amazon DCV サーバーでは、ファイルはセッションの作成時に実行されます。

タイプ: 文字列

必須: いいえ

`AutorunFileArguments`

Linux Amazon DCV サーバーの仮想セッションでサポートされています。Windows および Linux Amazon DCV サーバーのコンソールセッションではサポートされません。セッション内での実行時に AutorunFile に渡されるコマンドライン引数。引数は、所定の配列に表示される順序で渡されます。引数の最大許容数および各引数の最大許容長さを設定できます。詳細については、「Amazon DCV セッションマネージャー管理者ガイド」の「ブローカー設定ファイル」を参照してください。

タイプ: 文字列の配列

必須: いいえ

`DisableRetryOnFailure`

何らかの理由で Amazon DCV ホストで失敗した場合にセッション作成リクエストを再試行するかどうかを示します。セッション再試行メカニズムの作成の詳細については、「Amazon DCV セッションマネージャー管理者ガイド」の「ブローカー設定ファイル」を参照してください。

型: ブール値

デフォルト: false

必須: いいえ

`Requirements`

セッションを配置するためにサーバーで満たされなければならない要件。この要件にはサーバータグやサーバープロパティが含まれていることがあり、サーバータグとサーバープロパティはいずれも DescribeServers API の呼び出しにより取得されます。

要件の条件式:

- *a* != *b* *a* と *b* が等しくない場合は true

- *a* = *b* *a* と *b* が等しい場合は true
- *a* > *b* *a* が *b* よりも大きい場合は true
- *a* >= *b* *a* が *b* 以上である場合は true
- *a* < *b* *a* が *b* よりも小さい場合は true
- *a* <= *b* *a* が *b* 以下である場合は true
- *a* = *b* *a* に文字列 *b* が含まれている場合は true

要件のブール演算子:

- *a* and *b* *a* と *b* が true である場合は true
- *a* or *b* *a* か *b* が true である場合は true
- not *a* *a* が false である場合は true

タグキーには接頭辞として `tag:` を付け、サーバープロパティには接頭辞として `server:` を付ける必要があります。要件式では括弧 ( ) を使用できます。

要件の例:

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

`"server:Host.Memory.TotalBytes > 1024E6"` などの指数表現を使用して数値を指定できます。

サポートされているサーバープロパティは次のとおりです。

- `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`

タイプ: 文字列

必須: いいえ

`StorageRoot`

セッションストレージに使用されるフォルダのパスを指定します。Amazon DCV セッションストレージの詳細については、「Amazon DCV 管理者ガイド」の「セッションストレージの有効化」を参照してください。

タイプ: 文字列

必須: いいえ

レスポンスパラメータ

`Id`

セッションの一意の ID。

Name

セッション名。

Owner

セッション所有者。

Type

セッションのタイプ。

State

セッションの状態。リクエストが正常に完了すると、セッションが CREATING 状態に入ります。

Substate

セッションのサブステート。リクエストが正常に完了すると、サブステートが SESSION_PLACING サブステートに入ります。

例

Python

リクエスト

次の例では、3 つのセッションを作成します。

```
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),
])
```

レスポンス

以下は出力例です。

```
{
              "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

1 つ以上の Amazon DCV サーバーを記述します。

トピック

- リクエストパラメータ
- レスポンスパラメータ
- 例

リクエストパラメータ

ServerIds

記述する Amazon DCV サーバーの ID。ID を一切指定しない場合、すべてのサーバーがページ割り出力で返されます。

タイプ: 文字列の配列

必須: いいえ

NextToken

次の結果ページの取得に使用するトークン。

タイプ: 文字列

必須: いいえ

MaxResults

ページ割り出力でリクエストにより返される結果の最大数。このパラメータを使用すると、リクエストにより、指定した件数の結果と NextToken レスポンス要素が 1 ページにまとめられて返されます。最初のリクエストの残りの結果は、返された NextToken 値を含む別のリクエストを送信することで確認できます。

有効な範囲: 1 ~ 1000

デフォルト: 1000

タイプ: 整数

必須: いいえ

レスポンスパラメータ

RequestId

リクエストの一意の ID。

Servers

Amazon DCV サーバーに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Id

Amazon DCV サーバーの一意の ID。

Ip

Amazon DCV サーバーの IP アドレス。

Hostname

Amazon DCV サーバーのホスト名。

Endpoints

Amazon DCV サーバーのエンドポイントに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

IpAddress

サーバーエンドポイントの IP アドレス。

Port

サーバーエンドポイントのポート。

Protocol

サーバーエンドポイントで使用されるプロトコル。可能な値は以下のとおりです:

- `HTTP` — エンドポイントでは WebSocket (TCP) プロトコルが使用されます。
- `QUIC` — エンドポイントでは QUIC (UDP) プロトコルが使用されます。

WebUrlPath

サーバーエンドポイントのウェブ URL パス。HTTP プロトコルでのみ使用できます。

Version

Amazon DCV サーバーのバージョン。

SessionManagerAgentVersion

Amazon DCV サーバーで実行されている、バージョンのセッションマネージャーエージェント。

Availability

Amazon DCV サーバーの可用性。可能な値は以下のとおりです:

- `AVAILABLE` — サーバーは利用可能で、セッション配置の準備が整っています。
- `UNAVAILABLE` — サーバーは利用不能で、セッション配置を受け付けません。

UnavailabilityReason

Amazon DCV サーバーが利用不能である理由。可能な値は以下のとおりです:

- `SERVER_FULL` — Amazon DCV サーバーが実行可能な同時セッションの最大数に達しました。
- `SERVER_CLOSED` — CloseServer API を使用して Amazon DCV サーバーを利用不能にしました。
- `UNREACHABLE_AGENT` — セッションマネージャーブローカーが、Amazon DCV サーバーのセッションマネージャーエージェントと通信できません。
- `UNHEALTHY_DCV_SERVER` — セッションマネージャーエージェントが Amazon DCV サーバーと通信できません。
- `EXISTING_LOGGED_IN_USER` — (Windows Amazon DCV サーバのみ) ユーザーは現在 RDP を使用して Amazon DCV サーバーにログインしています。

- UNKNOWN — セッションマネージャーブローカーにより理由を特定できません。

ConsoleSessionCount

Amazon DCV サーバーのコンソールセッション数。

VirtualSessionCount

Amazon DCV サーバーの仮想セッション数。

Host

Amazon DCV サーバーが実行されているホストサーバーに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Os

ホストサーバーのオペレーティングシステムに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Family

オペレーティングシステムファミリー。可能な値は以下のとおりです:

- windows — ホストサーバーで Windows オペレーティングシステムが実行されています。
- linux — ホストサーバーで Linux オペレーティングシステムが実行されています。

Name

オペレーティングシステムの名前。

Version

オペレーティングシステムのバージョン。

KernelVersion

(Linux のみ) オペレーティングシステムのカーネルバージョン。

BuildNumber

(Windows のみ) オペレーティングシステムのビルド番号。

Memory

ホストサーバーのメモリに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

TotalBytes

ホストサーバーの総メモリ容量 (バイト)。

UsedBytes

ホストサーバーのメモリ使用量 (バイト)。

Swap

ホストサーバーのスワップファイルに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

TotalBytes

ホストサーバーのスワップファイルの合計サイズ (バイト)。

UsedBytes

ホストサーバーのスワップファイルの使用サイズ (バイト)。

Aws

Amazon EC2 インスタンスで実行されている Amazon DCV サーバーのみ。 AWS固有の情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Region

Amazon EC2 インスタンスの AWS リージョン。

Ec2InstanceType

Amazon EC2 インスタンスのタイプ。

Ec2InstanceId

Amazon EC2 インスタンスの ID。

Ec2ImageId

Amazon EC2 イメージの ID。

CpuInfo

ホストサーバーの CPU に関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Vendor

ホストサーバーの CPU のベンダー。

ModelName

ホストサーバーの CPU のモデル名。

Architecture

ホストサーバーの CPU のアーキテクチャ。

NumberOfCpus

ホストサーバーの CPU の数。

PhysicalCorePerCpu

CPU あたりの CPU コアの数。

CpuLoadAverage

ホストサーバーの CPU ロードに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

OneMinute

過去 1 分間の CPU ロードの平均。

FiveMinutes

過去 5 分間の CPU ロードの平均。

FifteenMinutes

過去 15 分間の CPU ロードの平均。

Gpus

ホストサーバーの GPU に関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Vendor

ホストサーバーの GPU のベンダー。

ModelName

ホストサーバーの GPU のモデル名。

LoggedInUsers

ホストサーバーに現在ログインしているユーザー。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Username

ログインしているユーザーのユーザー名。

Tags

サーバーに割り当てられているタグ。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Key

タグキー。

Value

タグ値。

例

Python

リクエスト

次の例では、使用可能なすべての Amazon DCV サーバーについて説明します。結果はページ割りされ、ページごとに結果が 2 件ずつ表示されます。

```
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)
```

レスポンス

以下は出力例です。

```
{
    "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

1 つ以上の Amazon DCV セッションを記述します。

トピック

- リクエストパラメータ
- レスポンスパラメータ
- 例

リクエストパラメータ

SessionIds

記述するセッションの ID。

タイプ: 文字列

必須: いいえ

NextToken

次の結果ページの取得に使用するトークン。

タイプ: 文字列

必須: いいえ

Filters

リクエストに適用する追加のフィルター。サポートされているフィルターは次のとおりです。

- tag:key — セッションに割り当てられたタグ。
- owner — セッションの所有者。

タイプ: 文字列

必須: いいえ

レスポンスパラメータ

Id

セッションの一意の ID。

Name

セッションの名前。

Owner

セッションの所有者。

Server

セッションが実行されているサーバーに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Ip

Amazon DCV サーバーホストの IP アドレス。

Hostname

Amazon DCV サーバーホストのホスト名。

Port

Amazon DCV サーバーが Amazon DCV クライアントと通信するポート。

Endpoints

Amazon DCV サーバーのエンドポイントに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

IpAddress

サーバーエンドポイントの IP アドレス。

Port

サーバーエンドポイントのポート。

Protocol

サーバーエンドポイントで使用されるプロトコル。可能な値は以下のとおりです:

- HTTP — エンドポイントでは WebSocket (TCP) プロトコルが使用されます。
- QUIC — エンドポイントでは QUIC (UDP) プロトコルが使用されます。

WebUrlPath

サーバーエンドポイントのウェブ URL パス。HTTP プロトコルでのみ使用できます。

Tags

サーバーに割り当てられているタグ。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Key

タグキー。

Value

タグ値。

Type

セッションのタイプ。

State

セッションの現在の状態。可能な値は以下のとおりです。

- CREATING — ブローカーでセッションが作成されています。
- READY — セッションはクライアント接続を受け入れる準備が整っている状態です。
- DELETING — セッションが削除されています。
- DELETED — セッションが削除されました。
- UNKNOWN — セッションの状態を判別できません。ブローカーとエージェントが通信できない可能性があります。

Substate

セッションの現在のサブステート。可能な値は以下のとおりです。

- SESSION_PLACING - セッションは利用可能な DCV サーバーに配置されるのを待っています。
- PENDING_PREPARATION - セッションは作成済みですが使用できません。DCV サーバーに接続されています。

CreationTime

セッションが作成された日時。

LastDisconnectionTime

クライアントが最後に切断された日時。

NumOfConnections

アクティブなクライアント接続の数。

StorageRoot

セッションストレージに使用されるフォルダのパスを指定します。Amazon DCV セッションストレージの詳細については、「Amazon DCV 管理者ガイド」の「セッションストレージの有効化」を参照してください。

タイプ: 文字列

必須: いいえ

例

Python

リクエスト

次の例では、user1 により所有されており os=windows のタグがあるセッションについて説明します。

```
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'}])
```

レスポンス

以下は出力例です。

```
{
    "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

指定された Amazon DCV セッションが消去され、ブローカーのキャッシュから削除されます。

トピック

- リクエストパラメータ
- レスポンスパラメータ
- 例

リクエストパラメータ

SessionId

削除するセッションの ID。

タイプ: 文字列

必須: はい

Owner

削除するセッションの所有者。

タイプ: 文字列

必須: はい

Force

Amazon DCV サーバーからセッションを削除しようとすると、そのセッションがブローカーのキャッシュから削除されます。これは、ブローカーのキャッシュから古いセッションを削除する場合に便利です。例えば、Amazon DCV サーバーが停止されたにもかかわらずセッションがブローカーに登録されている場合、このフラグを使用してブローカーのキャッシュからセッションを消去します。

それでもセッションがアクティブである場合は、ブローカーによって再キャッシュされることに注意してください。

有効な値: `true` | `false`

型: ブール値

必須: いいえ

レスポンスパラメータ

SessionId

セッションの ID

State

セッションが正常に削除された場合にのみ返されます。セッションの現在の状態を示します。リクエストが正常に完了すると、セッションが DELETING 状態に移行します。セッションが削除されるまでに数分かかることがあります。セッションは削除されると状態が DELETING から DELETED に移行します。

FailureReason

一部のセッションを削除できなかった場合にのみ返されます。セッションを削除できなかった理由を示します。

例

Python

リクエスト

次の例では 2 つのセッションを削除します。1 つは ID が SessionId123 であり user1 に所有されているセッションで、もう 1 つは ID が SessionIdabc で user99 に所有されているセッションです。

```
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')])
```

レスポンス

以下はサンプル出力です。SessionId123 は正常に削除されましたが、SessionIdabc は削除されませんでした。

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

GetSessionConnectionData

特定のユーザーによる特定の Amazon DCV セッションへの接続に関する接続情報を取得します。

トピック

- リクエストパラメータ
- レスポンスパラメータ
- 追加情報
- 例

リクエストパラメータ

SessionId

接続情報が表示されるセッションの ID。

タイプ: 文字列

必須: はい

User

接続情報が表示されるユーザーの名前。

タイプ: 文字列

必須: はい

レスポンスパラメータ

Id

セッションの一意の ID。

Name

セッションの名前。

Owner

セッションの所有者。

Server

セッションが実行されているサーバーに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Ip

Amazon DCV サーバーホストの IP アドレス。

Hostname

Amazon DCV サーバーホストのホスト名。

Port

Amazon DCV サーバーが Amazon DCV クライアントと通信するポート。

Endpoints

Amazon DCV サーバーのエンドポイントに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

IpAddress

サーバーエンドポイントの IP アドレス。

Port

サーバーエンドポイントのポート。

Protocol

サーバーエンドポイントで使用されるプロトコル。可能な値は以下のとおりです:

- `HTTP` — エンドポイントでは WebSocket (TCP) プロトコルが使用されます。
- `QUIC` — エンドポイントでは QUIC (UDP) プロトコルが使用されます。

WebUrlPath

サーバーエンドポイントのウェブ URL パス。HTTP プロトコルでのみ使用できます。

WebUrlPath

Amazon DCV サーバーの設定ファイルへのパス。

Tags

サーバーに割り当てられているタグ。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Key

タグキー。

Value

タグ値。

Type

セッションのタイプ。

State

セッションの現在の状態。可能な値は以下のとおりです。

- CREATING — ブローカーでセッションが作成されています。
- READY — セッションはクライアント接続を受け入れる準備が整っている状態です。
- DELETING — セッションが削除されています。
- DELETED — セッションが削除されました。
- UNKNOWN — セッションの状態を判別できません。ブローカーとエージェントが通信できない可能性があります。

CreationTime

セッションが作成された日時。

LastDisconnectionTime

クライアントが最後に切断された日時。

NumOfConnections

ユーザーが持つ同時セッション接続の数。

ConnectionToken

セッションへの接続に使用されている認証トークン。

追加情報

この API から取得した情報は、Amazon DCV セッションに接続するために Amazon DCV クライアントに渡すことができます。

Amazon DCV ウェブクライアントの場合は、ブラウザで開くことができる URL を作成できます。URL は以下の形式です。

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

Amazon DCV ネイティブクライアントの場合は、`dcv://` スキーマを使用して URL を作成できます。Amazon DCV ネイティブクライアントをインストールすると、`dcv://` URL のハンドラーとしてシステムに登録されます。URL は以下の形式です。

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

> **Note**
>
> Amazon EC2 を使用している場合、IP アドレスはパブリックアドレスである必要があります。ゲートウェイの背後に Amazon DCV ホストが設定されている場合は、SessionConnectionData API によって返されるアドレスではなく、ゲートウェイのアドレスを指定してください。

例

Python

リクエスト

次の例では、ユーザー名が `user1` であるユーザーと、ID が `sessionId12345` であるセッションに関する接続情報を取得します。

```
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')
```

レスポンス

以下は出力例です。

```
{
    "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

1 つ以上の Amazon DCV セッションのスクリーンショットを取得します。

イメージ形式を変更するには、Session Manager ブローカー設定で `session-screenshot-format`パラメータを設定します。「Amazon DCV セッションマネージャー管理者ガイド」の「ブローカー設定ファイル」を参照してください。

`GetSessionScreenshots` リクエストの `MaxWidth`または `MaxHeight`パラメータが指定されていない場合は、Session Manager ブローカー設定ファイルで設定された `session-screenshot-max-width`および `session-screenshot-max-height`値が使用されます。これらのパラメータを変更するには、「Amazon DCV セッションマネージャー管理者ガイド」の「ブローカー設定ファイル」も参照してください。

スクリーンショット解像度の上限値は、リモートセッション解像度に制限されます。`MaxWidth` および `MaxHeight`パラメータが現在のリモートセッション解像度よりも高い値に設定されている場合、結果のスクリーンショットは実際のセッション解像度に制限されます。

> **Note**
>
> アクセスコンソールからこれらの値を変更するには、「Amazon DCV アクセスコンソール管理者ガイド」の「ウェブクライアント設定ファイル」を参照してください。Session Manager CLI でこれらの値を変更するには、「Amazon DCV CLI ガイド`get-session-screenshots`」の「」を参照してください。

トピック

- リクエストパラメータ
- レスポンスパラメータ
- 例

リクエストパラメータ

`SessionId`

スクリーンショットの取得元となる Amazon DCV セッションの ID。

タイプ: 文字列

必須: はい

MaxWidth

セッションスクリーンショットの最大幅、ピクセル単位。指定しない場合、Session Manager ブローカー設定の値が適用されます。指定した場合、0 より大きい数値である必要があります。

タイプ: 整数

必須: いいえ

MaxHeight

セッションスクリーンショットの最大の高さをピクセル単位で表します。指定しない場合、Session Manager ブローカー設定の値が適用されます。指定した場合、0 より大きい数値である必要があります。

タイプ: 整数

必須: はい

レスポンスパラメータ

RequestId

リクエストの一意の ID。

SuccessfulList

成功したスクリーンショットに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

SessionScreenshot

スクリーンショットに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

SessionId

スクリーンショット撮影元である Amazon DCV セッションのセッション ID。

Images

イメージに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

Format

イメージの形式。可能値には `jpeg` や `png` などがあります。

Data

スクリーンショットイメージの base64 エンコード形式。

CreationTime

スクリーンショットが撮影された日時。

Primary

スクリーンショットが Amazon DCV セッションのプライマリディスプレイであるかどうかを示します。

UnsuccessfulList

失敗したスクリーンショットに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

GetSesionScreenshotRequestData

失敗した元のリクエスト。

SessionId

スクリーンショットの撮影元となる Amazon DCV セッションの ID。

FailureReason

失敗の理由。

GetSessionScreenshotRequestData

失敗した元のリクエスト。

例

Python

リクエスト

次の例では、最大幅を 800、最大高さを 600 に設定して、2 つのセッション (`sessionId1` および `sessionId2`) からスクリーンショットを取得します。セッション `sessionId2` は存在せず、失敗します。

```
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)
```

レスポンス

以下は出力例です。

```
{
    "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

1 つまたは複数の Amazon DCV サーバーを開きます。Amazon DCV サーバーでセッションを作成する前に、サーバーの状態を [開く] に変更する必要があります。Amazon DCV サーバーが [開く] になったら、サーバーで Amazon DCV セッションを作成できます。

トピック

- リクエストパラメータ
- レスポンスパラメータ
- 例

リクエストパラメータ

ServerId

起動するサーバーの ID。

タイプ: 文字列

必須: はい

レスポンスパラメータ

RequestId

リクエストの一意の ID。

SuccessfulList

正常に起動された Amazon DCV サーバーに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

ServerId

正常に起動されたサーバーの ID。

UnsuccessfulList

起動されなかった Amazon DCV サーバーに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

OpenServerRequestData

失敗した元のリクエストに関する情報。このデータ構造には、次のネスト済みレスポンスパラメータが含まれます。

ServerId

起動されなかった Amazon DCV サーバの ID。

FailureCode

失敗のコード。

FailureReason

失敗の理由。

例

Python

リクエスト

次の例では、2 つの Amazon DCV サーバー (`serverId1` と `serverId2`) を起動します。

```
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"])
```

レスポンス

以下は出力例です。

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

UpdateSessionPermissions

特定の Amazon DCV セッションのユーザーアクセス許可を更新します。

トピック

- リクエストパラメータ
- レスポンスパラメータ
- 例

リクエストパラメータ

SessionId

アクセス許可が変更されるセッションの ID を指定します。

タイプ: 文字列

必須: はい

Owner

アクセス許可が変更されるセッションの所有者。

タイプ: 文字列

必須: はい

PermissionFile

使用するアクセス許可ファイルの Base64 エンコードコンテンツ。詳細については、「Amazon DCV 管理者ガイド」の「Amazon DCV 認可の設定」を参照してください。

タイプ: 文字列

必須: はい

レスポンスパラメータ

SessionId

セッションの ID

例

Python

リクエスト

次の例では、セッション ID が `SessionId1897` であるセッションに新しいアクセス許可を設定します。

```
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')])
```

レスポンス

以下は出力例です。

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

Amazon DCV セッションマネージャーのリリースノートとドキュメント履歴

このページでは、Amazon DCV セッションマネージャーのリリースノートとドキュメント履歴を掲載します。

トピック

- Amazon DCV セッションマネージャーのリリースノート
- ドキュメント履歴

Amazon DCV セッションマネージャーのリリースノート

このセクションでは、Amazon DCV セッションマネージャーの大幅な更新、機能リリース、バグ修正の概要について説明します。更新はすべてリリース日別に整理されています。お客様からお寄せいただいたフィードバックに対応するために、ドキュメントを頻繁に更新しています。

トピック

- 2024.0-531 — 2025 年 6 月 17 日
- 2024.0-504 — 2025 年 3 月 31 日
- 2024.0-493 — 2025 年 1 月 15 日
- 2024.0-457 - 2024 年 10 月 1 日
- 2023.1-17652 - 2024 年 8 月 1 日
- 2023.1-16388 - 2024 年 6 月 26 日
- 2023.1— 2023 年 11 月 9 日
- 2023.0-15065— 2023 年 5 月 4 日
- 2023.0-14852— 2023 年 3 月 28 日
- 2022.2-13907— 2022 年 11 月 11 日
- 2022.1-13067— 2022 年 6 月 29 日
- 2022.0-11952 — 2022 年 2 月 23 日
- 2021.3-11591 — 2021 年 12 月 20 日
- 2021.2-11445 — 2021 年 11 月 18 日
- 2021.2-11190 — 2021 年 10 月 11 日

- 2021.2-11042 — 2021 年 9 月 1 日
- 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

ビルド番号	変更とバグ修正
• ブローカー: 531 • エージェント: 852 • CLI: 154	• 有効期限が切れる前に証明書を更新する機能を追加しました。 • NICE DCV を Amazon DCV にブランド変更しました。 • バグが修正されました。

2024.0-504 — 2025 年 3 月 31 日

ビルド番号	変更とバグ修正
• ブローカー: 504 • エージェント: 817 • CLI: 154	• AL2023 のサポートが追加されました。 • パフォーマンス向上とバグ修正が行われています。

2024.0-493 — 2025 年 1 月 15 日

ビルド番号	変更とバグ修正
• ブローカー: 493 • エージェント: 801 • CLI: 152	• スクリーンショットの最大の高さと幅を指定するパラメータを`GetSessionScreenshot` リクエストに追加しました。 • 到達できない Amazon DCV サーバーのセッションがシステムから削除されるまでの秒数を指定するパラメータをブローカー設定ファイルに追加しました。

ビルド番号	変更とバグ修正
	• ブローカー設定ファイルの `seconds-before-deleting-unreachable-dcv-server` パラメータが受け入れられない問題を修正しました。 • パフォーマンス向上とバグ修正が行われています。

2024.0-457 - 2024 年 10 月 1 日

ビルド番号	変更とバグ修正
• ブローカー: 457 • エージェント: 748 • CLI: 140	• NICE DCV を Amazon DCV にブランド変更しました。 • Ubuntu 24.04 に追加されたサポート。

2023.1-17652 - 2024 年 8 月 1 日

ビルド番号	変更とバグ修正
• ブローカー: 426 • エージェント: 748 • CLI: 140	• パフォーマンス向上とバグ修正が行われています。

2023.1-16388 - 2024 年 6 月 26 日

ビルド番号	変更とバグ修正
• ブローカー: 417 • エージェント: 748 • CLI: 140	• GB ではなく TB としてメモリが誤って表示されたバグを修正しました。 • パフォーマンス向上とバグ修正が行われています。

2023.1— 2023 年 11 月 9 日

ビルド番号	変更とバグ修正
• ブローカー: 410 • エージェント: 732 • CLI: 140	• バグを修正してパフォーマンスを改善しました。

2023.0-15065— 2023 年 5 月 4 日

ビルド番号	変更とバグ修正
• ブローカー: 392 • エージェント: 675 • CLI: 132	• ARM プラットフォームで Red Hat Enterprise Linux 9、Rocky Linux 9、CentOS Stream 9 のサポートが追加されました。

2023.0-14852— 2023 年 3 月 28 日

ビルド番号	変更とバグ修正
• ブローカー: 392 • エージェント: 642 • CLI: 132	• Red Hat Enterprise Linux 9、Rocky Linux 9、CentOS Stream 9 のサポートが追加されました。

2022.2-13907— 2022 年 11 月 11 日

ビルド番号	変更とバグ修正
• ブローカー: 382 • エージェント: 612 • CLI: 123	• `DescribeSessions` の応答に `Substate` フィールドが追加されました。

ビルド番号	変更とバグ修正
	• 使用中の URL に応じて CLI がブローカーに接続できない問題を修正しました。

2022.1-13067— 2022 年 6 月 29 日

ビルド番号	変更とバグ修正
• ブローカー: 355 • エージェント: 592 • CLI: 114	• Graviton AWS インスタンスでブローカーを実行するためのサポートが追加されました。 • Ubuntu 22.04 のエージェントとブローカーのサポートが追加されました。

2022.0-11952 — 2022 年 2 月 23 日

ビルド番号	変更とバグ修正
• ブローカー: 341 • エージェント: 520 • CLI: 112	• エージェントにログローテーション機能が追加されました。 • ブローカーに Java ホームを設定する設定パラメータを追加しました。 • ブローカーのキャッシュからディスクへのデータフラッシュが改善されました。 • CLI での URL 検証を修正しました。

2021.3-11591 — 2021 年 12 月 20 日

ビルド番号	新機能
• ブローカー: 307 • エージェント: 453 • CLI: 92	• Amazon DCV Connection Gateway との統合のサポートが追加されました。 • Ubuntu 18.04 と Ubuntu 20.04 のブローカーのサポートが追加されました。

2021.2-11445 — 2021 年 11 月 18 日

ビルド番号	変更とバグ修正
• ブローカー: 288 • エージェント: 413 • CLI: 54	• Windows ドメインを含むログイン名の検証に関する問題を修正しました。

2021.2-11190 — 2021 年 10 月 11 日

ビルド番号	変更とバグ修正
• ブローカー: 254 • エージェント: 413 • CLI: 54	• コマンドラインインターフェイスで Windows セッションを起動できない問題を修正しました。

2021.2-11042 — 2021 年 9 月 1 日

ビルド番号	新機能	変更とバグ修正
• ブローカー: 254 • エージェント: 413 • CLI: 37	• Amazon DCV セッションマネージャーで、コマンドラインインターフェイス (CLI) がサポートされるようになりました。API を呼び出すのではなく、CLI で、Amazon DCV セッションの作成と管理を実行できます。 • Amazon DCV セッションマネージャーにブローカーデータの永続性を導入しました。可用性を高めるために、ブローカーでは、サーバーの状態情報を外部データストアに残して	• 外部認可サーバーを登録するときに、認可サーバーで JSON 形式のウェブトークンの署名に使用されるアルゴリズムを指定できるようになりました。この変更により、Azure AD を外部認可サーバーとして使用できます。

ビルド番号	新機能	変更とバグ修正
	おいて、スタートアップ時にデータを復元することができます。	

2021.1-10557 — 2021 年 5 月 31 日

ビルド番号	新機能	変更とバグ修正
• ブローカー: 214 • エージェント: 365	• Amazon DCV セッションマネージャーにおいて、Linux で自動実行ファイルに渡される入力パラメータのサポートを追加しました。 • サーバープロパティを CreateSessions API に要件として渡すことができるようになりました。	• Windows での自動実行ファイルに関する問題を修正しました。

2021.0-10242 — 2021 年 4 月 12 日

ビルド番号	変更とバグ修正

• ブローカー: 183
• エージェント: 318
	• Amazon DCV セッションマネージャーに次の新しい API を導入しました。
	  • OpenServers
	  • CloseServers
	  • DescribeServers
	  • GetSessionScreenshots
	• 次の新しい設定パラメータも導入しました。
	  • ブローカーパラメータ: `session-screenshot-max-width` 、`session-screenshot-max-height` 、`session-screenshot-format` 、`create-sessions-queue-max-size` 、`create-sessions-queue-max-time-seconds`

ビルド番号	変更とバグ修正
	• エージェントパラメータ: `agent.autorun_folder` 、`max_virtual_sessions` 、`max_concurrent_sessions_per_user` エージェントパラメータ: `agent.autorun_folder` 、`max_virtual_sessions` 、`max_concurrent_sessions_per_user` エージェントパラメータ: `agent.autorun_folder` 、`max_virtual_sessions` 、`max_concurrent_sessions_per_user`

2020.2-9662 — 2020 年 12 月 4 日

ビルド番号	変更とバグ修正
• ブローカー: 114 • エージェント: 211	• 自動生成された TLS 証明書によってブローカーの起動が妨害される問題を修正しました。

2020.2-9508 — 2020 年 11 月 11 日

ビルド番号	変更とバグ修正
• ブローカー: 78 • エージェント: 183	• Amazon DCV セッションマネージャーの初回リリース。

ドキュメント履歴

次の表は、Amazon DCV セッションマネージャーの今回のリリースの内容をまとめたものです。

変更	説明	日付
Amazon DCV バージョン 2024.0-531	Amazon DCV セッションマネージャーは、Amazon DCV 2024.0-531 用に更新されました。詳細については、「???」を参照してください。	2025 年 6 月 17 日
Amazon DCV バージョン 2024.0-504	Amazon DCV セッションマネージャーは、Amazon DCV 2024.0-504 用に更新されました。詳細については、「???」を参照してください。	2025 年 3 月 31 日
Amazon DCV バージョン 2024.0-493	Amazon DCV セッションマネージャーは、Amazon DCV 2024.0-493 用に更新されました。詳細については、「2024.0-493 — 2025 年 1 月 15 日」を参照してください。	2025年 1 月 15 日
Amazon DCV バージョン 2024.0-457	Amazon DCV 2024.0-457 用に Amazon DCV セッションマネージャーが更新されました。詳細については、「2024.0-457 - 2024 年 10 月 1 日」を参照してください。	2024 年 9 月 30 日
Amazon DCV バージョン 2023.1-17652	Amazon DCV 2023.1-17652 用に Amazon DCV セッションマネージャーが更新されました。詳細については、「2023.1-17652 - 2024 年 8 月 1 日」を参照してください。	2024 年 8 月 1 日
Amazon DCV バージョン 2023.1-16388	Amazon DCV 2023.1-16388 用に Amazon DCV セッションマネージャーが更新されました。詳細については、「2023.1-16388 - 2024 年 6 月 26 日」を参照してください。	2024 年 6 月 26 日
Amazon DCV バー	Amazon DCV 2023.1 用に Amazon DCV セッションマネージャーが更新さ	2023 年 11 月 9 日

変更	説明	日付
ジョン 2023.1	れました。詳細については、「2023.1 — 2023 年 11 月 9 日」を参照してください。	
Amazon DCV バージョン 2023.0	Amazon DCV 2023.0 用に Amazon DCV セッションマネージャーが更新されました。詳細については、「2023.0-14852— 2023 年 3 月 28 日」を参照してください。	2023 年 3 月 28 日
Amazon DCV バージョン 2022.2	Amazon DCV 2022.2 用に Amazon DCV セッションマネージャーが更新されました。詳細については、「2022.2-13907— 2022 年 11 月 11 日」を参照してください。	2022 年 11 月 11 日
Amazon DCV バージョン 2022.1	Amazon DCV 2022.1 用に Amazon DCV セッションマネージャーが更新されました。詳細については、「2022.1-13067— 2022 年 6 月 29 日」を参照してください。	2022 年 1 月 29 日
Amazon DCV バージョン 2022.0	Amazon DCV 2022.0 用に Amazon DCV セッションマネージャーが更新されました。詳細については、「2022.0-11952 — 2022 年 2 月 23 日」を参照してください。	2022 年 2 月 23 日
Amazon DCV バージョン 2021.3	Amazon DCV 2021.3 用に Amazon DCV セッションマネージャーが更新されました。詳細については、「2021.3-11591 — 2021 年 12 月 20 日」を参照してください。	2021 年 12 月 20 日

変更	説明	日付
Amazon DCV バージョン 2021.2	Amazon DCV 2021.2 用に Amazon DCV セッションマネージャーが更新されました。詳細については、「2021.2-11042 — 2021 年 9 月 1 日」を参照してください。	2021 年 9 月 1 日
Amazon DCV バージョン 2021.1	Amazon DCV 2021.1 用に Amazon DCV セッションマネージャーが更新されました。詳細については、「2021.1-10557 — 2021 年 5 月 31 日」を参照してください。	2021 年 5 月 31 日
Amazon DCV バージョン 2021.0	Amazon DCV 2021.0 用に Amazon DCV セッションマネージャーが更新されました。詳細については、「2021.0-10242 — 2021 年 4 月 12 日」を参照してください。	2021 年 4 月 12 日
Amazon DCV セッションマネージャーの初回リリース。	このコンテンツの初版です。	2020 年 11 月 11 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。