aws

개발자 가이드

NICE DCV 세션 관리자

NICE DCV 세션 관리자: 개발자 가이드

Table of Contents

NICE DCV 세션 관리자란 무엇인가요?

NICE DCV 세션 관리자는 설치 가능한 소프트웨어 패키지(에이전트 및 브로커)와 애플리케이션 프로그래밍 인터페이스(API)의 모음으로, 개발자와 독립 소프트웨어 개발 판매 회사(ISV)가 NICE DCV 서버 전체에서 NICE DCV 세션의 수명 주기를 프로그래밍 방식으로 생성하고 관리하는 프런트엔드 애플리케이션을 쉽게 구축할 수 있도록 합니다.

이 안내서에서는 세션 관리자 API를 사용하여 NICE DCV 세션의 수명 주기를 관리하는 방법을 설명합니다. 세션 관리자 브로커 및 에이전트를 설치하고 구성하는 방법에 대한 자세한 내용은 NICE DCV 세션 관리자 관리 안내서를 참조하세요.

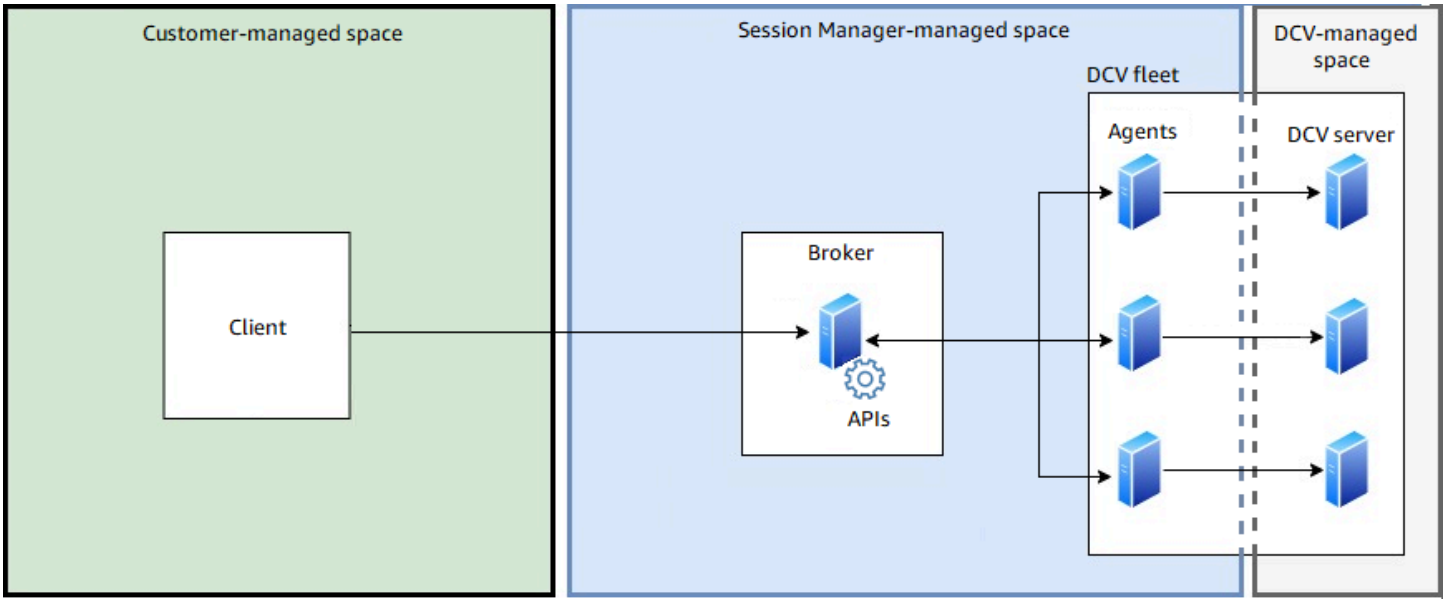
필수 조건

세션 관리자 API로 작업을 시작하려면 NICE DCV 및 NICE DCV 세션에 대해 알고 있어야 합니다. 자세한 내용은 NICE DCV 관리자 안내서를 참조하세요.

주제

세션 관리자의 작동 방식

다음 다이어그램은 세션 관리자의 종합적 구성 요소를 보여줍니다.



브로커

브로커는 세션 관리자 API를 호스팅하고 노출하는 웹 서버입니다. 브로커는 클라이언트로부터 NICE DCV 세션을 관리하기 위한 API 요청을 수신 및 처리한 다음 관련 에이전트에 지침을 전달합니다. 브로커는 NICE DCV 서버와 분리된 호스트에 설치해야 하지만 클라이언트가 브로커에 액세스할 수 있어야 하고 브로커가 에이전트에 액세스할 수 있어야 합니다.

에이전트

에이전트는 플릿의 각 NICE DCV 서버에 설치됩니다. 에이전트는 브로커로부터 지침을 받아 해당 NICE DCV 서버에서 실행합니다. 에이전트는 또한 NICE DCV 서버의 상태를 모니터링하고 정기적으로 상태 업데이트를 브로커에 보냅니다.

API

세션 관리자는 NICE DCV 서버 플릿에서 NICE DCV 세션을 관리하는 데 사용할 수 있는 REST API(애플리케이션 프로그래밍 인터페이스) 모음을 제공합니다. API는 브로커에서 호스팅되고 브로커에 의해 노출됩니다. 개발자는 API를 호출하는 사용자 지정 세션 관리 클라이언트를 구축할 수 있습니다.

클라이언트

클라이언트는 브로커가 표시하는 세션 관리자 API를 호출하기 위해 개발하는 프런트엔드 애플리케이션 또는 포털입니다. 최종 사용자는 클라이언트를 사용하여 플릿의 NICE DCV 서버에서 호스팅되는 세션을 관리합니다.

액세스 토큰

API 요청을 하려면 액세스 토큰을 제공해야 합니다. 등록된 클라이언트 API를 통해 브로커 또는 외부 권한 부여 서버에서 토큰을 요청할 수 있습니다. 토큰을 요청하고 액세스하려면 클라이언트 API가 유효한 보안 인증 정보를 제공해야 합니다.

클라이언트 API

클라이언트 API는 Swagger Codegen을 사용하여 세션 관리자 API 정의 YAML 파일에서 생성됩니다. 클라이언트 API는 API 요청을 하는 데 사용됩니다.

NICE DCV 세션

클라이언트가 연결할 수 있는 NICE DCV 서버에 NICE DCV 세션을 만들어야 합니다. 활성 세션이 있는 경우에만 클라이언트가 NICE DCV 서버에 연결할 수 있습니다. NICE DCV는 콘솔 및 가상 세션을 지원합니다. 세션 관리자 API를 사용하여 NICE DCV 세션의 수명 주기를 관리할 수 있습니다. NICE DCV 세션은 다음 상태 중 하나일 수 있습니다.

- `CREATING` - 브로커가 세션을 생성하는 중입니다.

- READY - 세션이 클라이언트 연결을 수락할 준비가 되었습니다.
- DELETING - 세션을 삭제 중입니다.
- DELETED - 세션이 삭제되었습니다.
- UNKNOWN - 세션 상태를 확인할 수 없습니다. 브로커와 에이전트가 통신하지 못할 수 있습니다.

기능

DCV 세션 관리자는 다음 기능을 제공합니다.

- NICE DCV 세션 정보 제공 - 여러 NICE DCV 서버에서 실행되는 세션에 대한 정보를 가져옵니다.
- 여러 NICE DCV 세션의 수명 주기 관리 - 한 번의 API 요청으로 여러 NICE DCV 서버에서 여러 사용자에 대한 여러 세션을 생성하거나 삭제합니다.
- 태그 지원 - 세션을 생성할 때 사용자 지정 태그를 사용하여 NICE DCV 서버 그룹을 대상으로 지정합니다.
- 여러 NICE DCV 세션에 대한 권한 관리 - 한 번의 API 요청으로 여러 세션에 대한 사용자 권한을 수정합니다.
- 연결 정보 제공 - NICE DCV 세션의 클라이언트 연결 정보를 검색합니다.
- 클라우드 및 온프레미스 지원 - AWS에서 온프레미스 또는 대체 클라우드 기반 서버와 함께 세션 관리자를 사용할 수 있습니다.

시작하기

이 섹션에서는 세션 관리자 API를 시작하는 방법을 설명합니다.

이 섹션에서는 `DescribeSessions` API를 예시로 사용하여 이 작업을 수행하는 방법을 살펴봅니다.

주제

- 클라이언트 API 생성
- API 클라이언트를 등록합니다.
- 액세스 토큰 가져오기 및 API 요청

클라이언트 API 생성

세션 관리자 API는 단일 YAML 파일에 정의되어 있습니다. API는 언어에 구애받지 않는 RESTful API에 대한 표준 인터페이스를 정의하는 OpenAPI3.0 사양을 기반으로 합니다. 자세한 내용은 OpenAPI 사양을 참조하세요.

YAML 파일을 사용하여 지원되는 언어 중 하나로 API 클라이언트를 생성할 수 있습니다. 이렇게 하려면 Swagger Codegen 3.0 이상을 사용해야 합니다. 지원되는 언어에 대한 자세한 내용은 swagger-codegen 리포지토리를 참조하세요.

API 클라이언트를 생성하려면

1. 세션 관리자 브로커에서 세션 관리자 API YAML 파일을 다운로드합니다. YAML 파일은 다음 URL에서 사용할 수 있습니다.

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

2. Swagger Codegen을 설치합니다.

   - macOS

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

   - 기타 플랫폼

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

```
$ cd swagger-codegen
```

3. API 클라이언트를 생성합니다.

   - macOS

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

   - 기타 플랫폼

   ```
   $ 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
   ```

API 클라이언트를 등록합니다.

API 요청을 하려면 먼저 브로커에서 액세스 토큰을 검색해야 합니다. 브로커에서 액세스 토큰을 가져오려면 브로커에 클라이언트 API의 보안 인증 정보를 제공해야 합니다. 보안 인증 정보는 클라이언트가 브로커에 등록될 때 생성되는 클라이언트 ID 및 클라이언트 암호를 기반으로 합니다. 클라이언트에 대한 클라이언트 ID와 클라이언트 암호가 없는 경우 브로커 관리자에게 요청해야 합니다. 클라이언트 API를 브로커에 등록하고 클라이언트 ID 및 암호를 얻는 방법에 대한 자세한 내용은 register-api-client를 참조하세요.

액세스 토큰 가져오기 및 API 요청

먼저 애플리케이션에 필요한 모델을 가져와야 합니다.

그런 다음 포트 번호(__PROTOCOL_HOST_PORT)를 포함하여 클라이언트 ID(__CLIENT_ID), 클라이언트 암호(__CLIENT_SECRET) 및 브로커 URL에 대한 변수를 선언합니다.

다음으로 클라이언트 보안 인증 정보를 생성하는 함수(build_client_credentials)를 만듭니다. 클라이언트 보안 인증 정보를 생성하려면 먼저 클라이언트 ID와 클라이언트 암호를 연결하고 콜론(*client_ID*:*client_password*)으로 값을 구분한 다음 전체 문자열을 Base64 형식으로 인코딩해야 합니다.

```
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')
```

이제 클라이언트 보안 인증 정보가 확보되었으므로 이를 사용하여 브로커에 액세스 토큰을 요청할 수 있습니다. 이렇게 하려면 get_access_token 함수를 만듭니다. https://*Broker_IP*:*8443*/oauth2/token?grant_type=client_credentials에 대해 POST를 호출하고 Basic으로 인코딩된 클라이언트 보안 인증 정보 및 application/x-www-form-urlencoded 콘텐츠 유형을 포함하는 권한 부여 헤더를 제공해야 합니다.

```
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
```

이제 클라이언트 API를 인스턴스화하는 데 필요한 함수를 만듭니다. 클라이언트 API를 인스턴스화하려면 클라이언트 구성과 요청에 사용할 헤더를 지정해야 합니다. 이 `get_client_configuration` 함수는 브로커의 IP 주소 및 포트, 브로커 관리자로부터 받았어야 하는 브로커의 자체 서명 인증서 경로를 포함하는 구성 객체를 생성합니다. 이 `set_request_headers` 함수는 클라이언트 보안 인증 정보와 액세스 토큰이 포함된 요청 헤더 객체를 만듭니다.

```
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
```

마지막으로 `DescribeSessions` API를 호출하는 기본 메서드를 만듭니다. 자세한 내용은 DescribeSessions 섹션을 참조하세요.

```
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'}])
```

세션 관리자 API 참조

이 섹션에는 각 세션 관리자 API 작업에 대한 설명, 구문 및 사용 예제가 나와 있습니다.

주제

CloseServers

하나 이상의 NICE DCV 서버를 종료합니다. NICE DCV 서버를 종료하면 NICE DCV 세션 배치에 해당 서버를 사용할 수 없게 됩니다. 종료된 서버에서는 NICE DCV 세션을 생성할 수 없습니다. 서버를 종료하면 해당 서버에서 어떠한 세션도 실행되지 않고 사용자가 새 세션을 만들 수 없게 됩니다.

주제

요청 파라미터

`ServerId`

종료할 서버의 ID입니다.

유형: 문자열

필수 항목 여부: 예

Force

종료 작업을 강제로 실행합니다. `true` 값을 지정하면 실행 중인 세션이 있더라도 서버가 종료됩니다. 세션은 계속 실행됩니다.

유형: 부울

필수 항목 여부: 아니요

응답 파라미터

RequestId

요청의 고유 ID입니다.

SuccessfulList

성공적으로 종료된 NICE DCV 서버에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

ServerId

성공적으로 종료된 서버의 ID입니다.

UnsuccessfulList

종료할 수 없는 NICE DCV 서버에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

CloseServerRequestData

실패한 원래 요청에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

ServerId

종료할 수 없는 NICE DCV 서버의 ID입니다.

Force

요청된 강제 파라미터입니다.

FailureCode

실패 코드입니다.

FailureReason

실패 이유

예

Python

요청

다음 예제에서는 두 개의 NICE DCV 서버(serverId1 및 serverId2)를 종료합니다. serverId2 서버가 존재하지 않아 오류가 발생합니다.

```
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"])
```

응답

다음은 출력 샘플입니다.

```
{
    "RequestId": "4d7839b2-a03c-4b34-a40d-06c8b21099e6",
    "SuccessfulList": [
        {
```

```
            "ServerId": "serverId1"
        }
    ],
    "UnsuccessfulList": [
        {
            "OpenServerRequestData": {
                "ServerId": "serverId2"
            },
            "FailureCode": "DCV_SERVER_NOT_FOUND",
            "FailureReason": "Dcv server not found."
        }
    ]
}
```

CreateSessions

지정된 세부 정보로 새 NICE DCV 세션을 생성합니다.

API 작업

- 요청 파라미터
- 응답 파라미터
- 예

요청 파라미터

`Name`

세션의 이름입니다.

유형: 문자열

필수 항목 여부: 예

`Owner`

세션 소유자의 이름입니다. 이는 대상 NICE DCV 서버의 기존 사용자 이름이어야 합니다.

유형: 문자열

필수 항목 여부: 예

`Type`

세션 유형입니다. 세션 유형에 대한 자세한 내용은 NICE DCV 관리자 안내서의 NICE DCV 세션 소개를 참조하세요.

유효한 값: CONSOLE | VIRTUAL

유형: 문자열

필수 항목 여부: 예

`InitFile`

Linux NICE DCV 서버의 가상 세션에서 지원됩니다. Windows 및 Linux NICE DCV 서버의 콘솔 세션에서는 지원되지 않습니다. 세션을 만들 때 세션을 초기화하기 위해 실행할 NICE DCV 서버의 사용자 지정 스크립트 경로입니다. 파일 경로는 `agent.init_folder` 에이전트 구성 파라미터에 지정된 init 디렉터리를 기준으로 합니다. 지정된 init 디렉터리에 파일이 있는 경우 파일 이름만 지정하세요. 지정된 init 디렉터리에 파일이 없는 경우 상대 경로를 지정하세요. 자세한 내용은 NICE DCV 세션 관리자 관리 안내서의 에이전트 구성 파일을 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

`MaxConcurrents`

동시에 접속할 수 있는 NICE DCV 클라이언트의 최대 수입니다.

유형: 정수

필수 항목 여부: 아니요

`DcvGlEnabled`

가상 세션이 하드웨어 기반 OpenGL을 사용하도록 구성되었는지 여부를 나타냅니다. 가상 세션에서만 지원됩니다. 이 파라미터는 Windows NICE DCV 서버에서는 지원되지 않습니다.

유효한 값: true | false

유형: 부울

필수 항목 여부: 아니요

`PermissionsFile`

권한 파일의 Base64로 인코딩된 내용입니다. 생략할 경우 서버 기본값이 기본 설정됩니다. 자세한 내용은 NICE DCV 관리자 안내서의 NICE DCV 권한 부여 구성을 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

`EnqueueRequest`

요청을 즉시 이행할 수 없는 경우 요청을 대기열에 넣을지 여부를 나타냅니다.

유형: 부울

기본값: false

필수 항목 여부: 아니요

`AutorunFile`

Windows NICE DCV 서버의 콘솔 세션과 Linux NICE DCV 서버의 가상 세션에서 지원됩니다. Linux NICE DCV 서버의 콘솔 세션에서는 지원되지 않습니다.

세션 내에서 실행될 호스트 서버의 파일 경로입니다. 파일 경로는 `agent.autorun_folder` 에이전트 구성 파라미터에 지정된 autorun 디렉터리를 기준으로 합니다. 지정된 autorun 디렉터리에 파일이 있는 경우 파일 이름만 지정하세요. 지정된 autorun 디렉터리에 파일이 없는 경우 상대 경로를 지정하세요. 자세한 내용은 NICE DCV 세션 관리자 관리 안내서의 에이전트 구성 파일을 참조하세요.

파일은 지정된 소유자를 대신하여 실행됩니다. 지정된 소유자에게는 서버에서 파일을 실행할 수 있는 권한이 있어야 합니다. Windows NICE DCV 서버에서는 소유자가 세션에 로그인할 때 파일이 실행됩니다. Linux NICE DCV 서버에서는 세션이 생성될 때 파일이 실행됩니다.

유형: 문자열

필수 항목 여부: 아니요

`AutorunFileArguments`

Linux NICE DCV 서버의 가상 세션에서 지원됩니다. Windows 및 Linux NICE DCV 서버의 콘솔 세션에서는 지원되지 않습니다. 세션 내에서 실행될 때 AutoRunFile에 전달되는 명령줄 인수입니다. 인수는 지정된 배열에 나타나는 순서대로 전달됩니다. 허용되는 최대 인수 개수와 각 인수의 최대

허용 길이를 구성할 수 있습니다. 자세한 내용은 NICE DCV 세션 관리자 관리 안내서의 브로커 구성 파일을 참조하세요.

유형: 문자열 배열

필수 항목 여부: 아니요

`DisableRetryOnFailure`

어떤 이유로든 NICE DCV 호스트에서 세션 생성 요청이 실패한 후 다시 시도하지 않을 것인지 여부를 나타냅니다. 세션 재시도 생성 메커니즘에 대한 자세한 내용은 NICE DCV 세션 관리자 관리 안내서의 브로커 구성 파일을 참조하세요.

유형: 부울

기본값: false

필수 항목 여부: 아니요

`Requirements`

세션을 배치하기 위해 서버가 충족해야 하는 요구 사항입니다. 요구 사항에는 서버 태그 및/또는 서버 속성이 포함될 수 있으며, 서버 태그와 서버 속성은 모두 DescribeServers API를 호출하여 검색됩니다.

요구 사항 조건 표현식:

- *a* != *b* - *a*와 *b*가 같지 않은 경우 true
- *a* = *b* - *a*와 *b*가 같은 경우 true
- *a* > *b* - *a*가 *b*보다 큰 경우 true
- *a* >= *b* - *a*가 *b*보다 크거나 같은 경우 true
- *a* < *b* - *a*가 *b*보다 작은 경우 true
- *a* <= *b* - *a*가 *b*보다 작거나 같은 경우 true
- *a* = *b* - *a*에 *b*가 포함된 경우 true

요구 사항 부울 연산자:

- *a* and *b* - *a*와 *b*가 참인 경우 true
- *a* or *b* - *a* 또는 *b* 가 참인 경우 true
- not *a* - *a*가 거짓인 경우 true

태그 키에는 tag: 접두사를 붙이고 서버 속성에는 server: 접두사를 붙여야 합니다. 요구 사항 표현식에는 괄호(( ))가 지원됩니다.

요구 사항 예제:

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

지수 표기법을 사용하여 숫자 값을 지정할 수 있습니다(예: `"server:Host.Memory.TotalBytes > 1024E6"`).

지원되는 서버 속성은 다음과 같습니다.

- `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`

유형: 문자열

필수 항목 여부: 아니요

`StorageRoot`

세션 스토리지에 사용되는 폴더 경로를 지정합니다. NICE DCV 세션 스토리지에 대한 자세한 내용은 NICE DCV 관리자 안내서의 세션 스토리지 활성화를 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

응답 파라미터

`Id`

세션의 고유 ID입니다.

`Name`

세션 이름

`Owner`

세션 소유자입니다.

`Type`

세션 유형입니다.

`State`

세션의 상태입니다. 요청이 성공적으로 완료되면 세션이 `CREATING` 상태로 전환됩니다.

Substate

세션의 하위 상태입니다. 요청이 성공적으로 완료되면 세션이 SESSION_PLACING 하위 상태로 전환됩니다.

예

Python

요청

다음 예제에서는 세 개의 세션을 만듭니다.

```
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),
])
```

응답

다음은 출력 샘플입니다.

```
{
              "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

하나 이상의 NICE DCV 서버를 설명합니다.

주제

요청 파라미터

`ServerIds`

설명해야 할 NICE DCV 서버의 ID입니다. ID가 지정되지 않으면 모든 서버가 페이지 매김 출력으로 반환됩니다.

유형: 문자열 배열

필수 항목 여부: 아니요

`NextToken`

결과의 다음번 페이지를 가져오기 위한 토큰입니다.

유형: 문자열

필수 항목 여부: 아니요

`MaxResults`

페이지가 매겨진 출력의 요청에서 반환되는 결과의 최대 수입니다. 이 파라미터를 사용하면 요청은 `NextToken` 응답 요소와 함께 단일 페이지에서 지정된 수의 결과만 반환합니다. 반환된 `NextToken` 값과 함께 다른 요청을 보내면 초기 요청의 나머지 결과를 확인할 수 있습니다.

유효 범위: 1~1,000

기본값: 1000

유형: 정수

필수 항목 여부: 아니요

응답 파라미터

`RequestId`

요청의 고유 ID입니다.

`Servers`

NICE DCV 서버에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

`Id`

NICE DCV 서버의 고유 ID입니다.

`Ip`

NICE DCV 서버의 IP 주소입니다.

`Hostname`

NICE DCV 서버의 호스트 이름입니다.

`Endpoints`

NICE DCV 서버 엔드포인트에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

`IpAddress`

서버 엔드포인트의 IP 주소입니다.

`Port`

서버 엔드포인트의 포트입니다.

`Protocol`

서버 엔드포인트가 사용하는 프로토콜입니다. 가능한 값은 다음과 같습니다.

- `HTTP` - 엔드포인트가 WebSocket(TCP) 프로토콜을 사용합니다.
- `QUIC` - 엔드포인트가 QUIC(UDP) 프로토콜을 사용합니다.

`WebUrlPath`

서버 엔드포인트의 웹 URL 경로입니다. HTTP 프로토콜에만 사용할 수 있습니다.

`Version`

NICE DCV 서버의 버전입니다.

SessionManagerAgentVersion

NICE DCV 서버에서 실행되는 세션 관리자 에이전트 버전입니다.

Availability

NICE DCV 서버의 가용성입니다. 가능한 값은 다음과 같습니다.

- AVAILABLE - 서버를 사용할 수 있으며 세션을 배치할 준비가 되었습니다.
- UNAVAILABLE - 서버를 사용할 수 없으며 세션 배치를 수락할 수 없습니다.

UnavailabilityReason

NICE DCV 서버를 사용할 수 없는 이유입니다. 가능한 값은 다음과 같습니다.

- SERVER_FULL - NICE DCV 서버가 실행할 수 있는 최대 동시 세션 수에 도달했습니다.
- SERVER_CLOSED - CloseServer API를 사용하여 NICE DCV 서버를 사용할 수 없게 되었습니다.
- UNREACHABLE_AGENT - 세션 관리자 브로커가 NICE DCV 서버의 세션 관리자 에이전트와 통신할 수 없습니다.
- UNHEALTHY_DCV_SERVER - 세션 관리자 에이전트가 NICE DCV 서버와 통신할 수 없습니다.
- EXISTING_LOGGED_IN_USER - (Windows NICE DCV 서버만 해당) 사용자가 현재 RDP를 사용하여 NICE DCV 서버에 로그인되어 있습니다.
- UNKNOWN - 세션 관리자 브로커가 이유를 확인할 수 없습니다.

ConsoleSessionCount

NICE DCV 서버의 콘솔 세션 수입니다.

VirtualSessionCount

NICE DCV 서버의 가상 세션 수입니다.

Host

NICE DCV 서버가 실행되고 있는 호스트 서버에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Os

호스트 서버 운영 체제에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Family

운영 체제 제품군입니다. 가능한 값은 다음과 같습니다.

- `windows` - 호스트 서버가 Windows 운영 체제를 실행 중입니다.
- `linux` - 호스트 서버가 Linux 운영 체제를 실행 중입니다.

Name

운영 체제 이름.

Version

운영 체제 버전.

KernelVersion

(Linux만 해당) 운영 체제의 커널 버전입니다.

BuildNumber

(Windows만 해당) 운영 체제의 빌드 번호입니다.

Memory

호스트 서버 메모리에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

TotalBytes

호스트 서버의 총 메모리(바이트)입니다.

UsedBytes

호스트 서버에서 사용된 메모리(바이트)입니다.

Swap

호스트 서버의 스왑 파일에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

TotalBytes

호스트 서버의 총 스왑 파일 크기(바이트)입니다.

UsedBytes

호스트 서버에서 사용된 스왑 파일 크기(바이트)입니다.

Aws

Amazon EC2 인스턴스에서 실행되는 NICE DCV 서버에만 해당됩니다. AWS 특정 정보를 나타냅니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Region

Amazon EC2 인스턴스의 AWS 리전입니다.

Ec2InstanceType

Amazon EC2 인스턴스 유형입니다.

Ec2InstanceId

Amazon EC2 인스턴스의 ID입니다.

Ec2ImageId

Amazon EC2 이미지의 ID입니다.

CpuInfo

호스트 서버 CPU에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Vendor

호스트 서버 CPU의 공급업체입니다.

ModelName

호스트 서버 CPU의 모델 이름입니다.

Architecture

호스트 서버 CPU의 아키텍처입니다.

NumberOfCpus

호스트 서버의 CPU 개수입니다.

PhysicalCorePerCpu

CPU당 CPU 코어 수입니다.

CpuLoadAverage

호스트 서버 CPU 부하에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

OneMinute

마지막 1분 동안의 평균 CPU 부하입니다.

FiveMinutes

마지막 5분 동안의 평균 CPU 부하입니다.

FifteenMinutes

마지막 15분 동안의 평균 CPU 부하입니다.

Gpus

호스트 서버 GPU에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Vendor

호스트 서버 GPU의 공급업체입니다.

ModelName

호스트 서버 GPU의 모델 이름입니다.

LoggedInUsers

현재 호스트 서버에 로그인한 사용자입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Username

로그인한 사용자의 사용자 이름입니다.

Tags

서버에 할당된 태그입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Key

태그 키.

Value

태그 값.

예

Python

요청

다음 예제에서는 사용 가능한 모든 NICE DCV 서버에 대해 설명합니다. 결과는 페이지당 두 개의 결과를 표시하도록 페이지가 매겨집니다.

```
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)
```

응답

다음은 출력 샘플입니다.

```
{
    "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

하나 이상의 NICE DCV 세션을 설명합니다.

주제

- 요청 파라미터
- 응답 파라미터
- 예

요청 파라미터

`SessionIds`

설명할 세션의 ID입니다.

유형: 문자열

필수 항목 여부: 아니요

NextToken

결과의 다음번 페이지를 가져오기 위한 토큰입니다.

유형: 문자열

필수 항목 여부: 아니요

Filters

요청에 적용할 추가 필터입니다. 지원되는 필터는 다음과 같습니다.

- tag:key - 세션에 할당된 태그입니다.
- owner - 세션 소유자입니다.

유형: 문자열

필수 항목 여부: 아니요

응답 파라미터

Id

세션의 고유 ID입니다.

Name

세션의 이름입니다.

Owner

세션 소유자입니다.

Server

세션이 실행되고 있는 서버의 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Ip

NICE DCV 서버 호스트의 IP 주소입니다.

Hostname

NICE DCV 서버 호스트의 호스트 이름입니다.

Port

NICE DCV 서버가 NICE DCV 클라이언트와 통신하는 데 사용되는 포트입니다.

Endpoints

NICE DCV 서버 엔드포인트에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

IpAddress

서버 엔드포인트의 IP 주소입니다.

Port

서버 엔드포인트의 포트입니다.

Protocol

서버 엔드포인트가 사용하는 프로토콜입니다. 가능한 값은 다음과 같습니다.

- `HTTP` - 엔드포인트가 WebSocket(TCP) 프로토콜을 사용합니다.
- `QUIC` - 엔드포인트가 QUIC(UDP) 프로토콜을 사용합니다.

WebUrlPath

서버 엔드포인트의 웹 URL 경로입니다. HTTP 프로토콜에만 사용할 수 있습니다.

Tags

서버에 할당된 태그입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Key

태그 키.

Value

태그 값.

Type

세션 유형입니다.

State

세션의 현재 상태입니다. 가능한 값은 다음과 같습니다.

- CREATING - 브로커가 세션을 생성하는 중입니다.
- READY - 세션이 클라이언트 연결을 수락할 준비가 되었습니다.
- DELETING - 세션을 삭제 중입니다.
- DELETED - 세션이 삭제되었습니다.
- UNKNOWN - 세션 상태를 확인할 수 없습니다. 브로커와 에이전트가 통신하지 못할 수 있습니다.

Substate

세션의 현재 하위 상태입니다. 가능한 값은 다음과 같습니다.

- SESSION_PLACING - 세션이 사용 가능한 DCV 서버에 배치되기를 기다리고 있습니다.
- PENDING_PREPARATION - 세션이 생성되었지만 사용할 수 없으며 DCV 서버에 연결되어 있습니다.

CreationTime

세션이 생성된 날짜와 시간입니다.

LastDisconnectionTime

클라이언트의 마지막 연결 해제 날짜 및 시간입니다.

NumOfConnections

활성 클라이언트 연결 수입니다.

StorageRoot

세션 스토리지에 사용되는 폴더 경로를 지정합니다. NICE DCV 세션 스토리지에 대한 자세한 내용은 NICE DCV 관리자 안내서의 세션 스토리지 활성화를 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

예

Python

요청

다음 예제에서는 user1에서 소유하고 os=windows 태그가 있는 세션에 대해 설명합니다.

```
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'}])
```

응답

다음은 출력 샘플입니다.

```
{
    "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

지정된 NICE DCV 세션을 삭제하고 브로커의 캐시에서 해당 세션을 제거합니다.

주제

요청 파라미터

`SessionId`

삭제할 세션의 ID입니다.

유형: 문자열

필수 항목 여부: 예

`Owner`

삭제할 세션의 소유자입니다.

유형: 문자열

필수 항목 여부: 예

Force

NICE DCV 서버에서 삭제를 시도하여 브로커 캐시에서 세션을 제거합니다. 이는 브로커 캐시에서 오래된 세션을 제거하는 데 유용합니다. 예를 들어 NICE DCV 서버가 중지되었지만 세션이 여전히 브로커에 등록되어 있는 경우, 이 플래그를 사용하여 브로커 캐시에서 세션을 제거하세요.

세션이 여전히 활성 상태인 경우 브로커가 다시 캐싱한다는 점에 유의하세요.

유효한 값: `true` | `false`

유형: 부울

필수 항목 여부: 아니요

응답 파라미터

SessionId

세션 ID

State

세션이 성공적으로 삭제된 경우에만 반환됩니다. 세션의 현재 상태를 나타냅니다. 요청이 성공적으로 완료되면 세션이 `DELETING` 상태로 전환됩니다. 세션이 삭제되는 데 몇 분 정도 걸릴 수 있습니다. 세션이 삭제되면 상태가 `DELETING`에서 `DELETED`로 전환됩니다.

FailureReason

일부 세션을 삭제할 수 없는 경우에만 반환됩니다. 세션을 삭제할 수 없는 이유를 나타냅니다.

예

Python

요청

다음 예제에서는 두 개의 세션, 즉 `user1`이 소유한 `SessionId123` ID의 세션과 `user99`가 소유한 `SessionIdabc` ID의 세션을 삭제합니다.

```
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')])
```

응답

다음은 출력 샘플입니다. SessionId123은 성공적으로 삭제되었지만 SessionIdabc는 삭제할 수 없습니다.

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

GetSessionConnectionData

특정 NICE DCV 세션에 대한 특정 사용자 연결에 대한 연결 정보를 가져옵니다.

주제

요청 파라미터

`SessionId`

연결 정보를 볼 세션의 ID입니다.

유형: 문자열

필수 항목 여부: 예

`User`

연결 정보를 보려는 사용자의 이름입니다.

유형: 문자열

필수 항목 여부: 예

응답 파라미터

`Id`

세션의 고유 ID입니다.

`Name`

세션의 이름입니다.

Owner

세션 소유자입니다.

Server

세션이 실행되고 있는 서버의 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Ip

NICE DCV 서버 호스트의 IP 주소입니다.

Hostname

NICE DCV 서버 호스트의 호스트 이름입니다.

Port

NICE DCV 서버가 NICE DCV 클라이언트와 통신하는 데 사용되는 포트입니다.

Endpoints

NICE DCV 서버 엔드포인트에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

IpAddress

서버 엔드포인트의 IP 주소입니다.

Port

서버 엔드포인트의 포트입니다.

Protocol

서버 엔드포인트가 사용하는 프로토콜입니다. 가능한 값은 다음과 같습니다.

- `HTTP` - 엔드포인트가 WebSocket(TCP) 프로토콜을 사용합니다.
- `QUIC` - 엔드포인트가 QUIC(UDP) 프로토콜을 사용합니다.

WebUrlPath

서버 엔드포인트의 웹 URL 경로입니다. HTTP 프로토콜에만 사용할 수 있습니다.

WebUrlPath

NICE DCV 서버의 구성 파일 경로입니다.

Tags

서버에 할당된 태그입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Key

태그 키.

Value

태그 값.

Type

세션 유형입니다.

State

세션의 현재 상태입니다. 가능한 값은 다음과 같습니다.

- CREATING - 브로커가 세션을 생성하는 중입니다.
- READY - 세션이 클라이언트 연결을 수락할 준비가 되었습니다.
- DELETING - 세션을 삭제 중입니다.
- DELETED - 세션이 삭제되었습니다.
- UNKNOWN - 세션 상태를 확인할 수 없습니다. 브로커와 에이전트가 통신하지 못할 수 있습니다.

CreationTime

세션이 생성된 날짜와 시간입니다.

LastDisconnectionTime

클라이언트의 마지막 연결 해제 날짜 및 시간입니다.

NumOfConnections

세션에 대해 사용자에게 주어진 동시 연결 수입니다.

ConnectionToken

세션 연결에 사용할 인증 토큰입니다.

추가 정보

이 API에서 얻은 정보는 NICE DCV 세션에 연결하기 위해 NICE DCV 클라이언트로 전달될 수 있습니다.

NICE DCV 웹 클라이언트의 경우 브라우저에서 열 수 있는 URL을 구축할 수 있습니다. URL의 형식은 다음과 같습니다.

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

NICE DCV 네이티브 클라이언트의 경우 dcv:// 스키마를 사용하여 URL을 작성할 수 있습니다. NICE DCV 네이티브 클라이언트가 설치되면 해당 클라이언트가 시스템에 dcv:// URL 핸들러로 등록됩니다. URL의 형식은 다음과 같습니다.

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

> **Note**
>
> Amazon EC2를 사용하는 경우 IP 주소는 퍼블릭 IP 주소여야 합니다. 구성에서 게이트웨이 뒤에 NICE DCV 호스트가 있는 경우 SessionConnectionData API에서 반환되는 주소 대신 게이트웨이 주소를 지정하세요.

예

Python

요청

다음 예제에서는 사용자 이름이 user1인 사용자와 ID가 sessionId12345인 세션의 연결 정보를 가져옵니다.

```
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')
```

응답

다음은 출력 샘플입니다.

```
{
    "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

하나 이상의 NICE DCV 세션 스크린샷을 가져옵니다.

스크린샷의 이미지 파일 유형과 해상도는 세션 관리자 브로커 구성에 따라 달라집니다. 이미지 파일 유형을 수정하려면 `session-screenshot-format` 파라미터를 구성하세요. 해상도를 수정하려면 `session-screenshot-max-width` 및 `session-screenshot-max-height` 파라미터를 구성하세요. 자세한 내용은 NICE DCV 세션 관리자 관리 안내서의 브로커 구성 파일을 참조하세요.

주제

요청 파라미터

`SessionId`

스크린샷을 가져올 NICE DCV 세션의 ID입니다.

유형: 문자열

필수 항목 여부: 예

응답 파라미터

`RequestId`

요청의 고유 ID입니다.

`SuccessfulList`

성공적인 스크린샷에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

`SessionScreenshot`

스크린샷에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

SessionId

스크린샷이 촬영된 NICE DCV 세션의 ID입니다.

Images

이미지에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

Format

이미지의 형식입니다. 가능한 값은 jpeg 및 png입니다.

Data

스크린샷 이미지는 base64로 인코딩된 형식입니다.

CreationTime

스크린샷이 촬영된 날짜 및 시간입니다.

Primary

스크린샷이 NICE DCV 세션의 기본 디스플레이인지 여부를 나타냅니다.

UnsuccessfulList

실패한 스크린샷에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

GetSesionScreenshotRequestData

실패한 원래 요청입니다.

SessionId

스크린샷이 촬영되어야 했던 NICE DCV 세션의 ID입니다.

FailureReason

실패 이유

예

Python

요청

다음 예제에서는 두 세션(sessionId1 및 sessionId2)에서 스크린샷을 가져옵니다. sessionId2 세션이 존재하지 않아 오류가 발생합니다.

```
from swagger_client.models.describe_servers_request_data import
 DescribeServersRequestData

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

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

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

응답

다음은 출력 샘플입니다.

```
{
    "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

하나 이상의 NICE DCV 서버를 엽니다. NICE DCV 서버에서 NICE DCV 세션을 생성하려면 먼저 서버의 상태를 열린 상태로 변경해야 합니다. NICE DCV 서버가 열리면 서버에서 NICE DCV 세션을 생성할 수 있습니다.

주제

요청 파라미터

`ServerId`

열려는 서버의 ID입니다.

유형: 문자열

필수 항목 여부: 예

응답 파라미터

`RequestId`

요청의 고유 ID입니다.

SuccessfulList

성공적으로 열린 NICE DCV 서버에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

ServerId

성공적으로 열린 서버의 ID입니다.

UnsuccessfulList

열 수 없는 NICE DCV 서버에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

OpenServerRequestData

실패한 원래 요청에 대한 정보입니다. 이 데이터 구조에는 다음과 같은 중첩된 응답 파라미터가 포함됩니다.

ServerId

열 수 없는 NICE DCV 서버의 ID입니다.

FailureCode

실패 코드입니다.

FailureReason

실패 이유

예

Python

요청

다음 예제에서는 두 개의 NICE DCV 서버(`serverId1` 및 `serverId2`)를 엽니다.

```
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"])
```

응답

다음은 출력 샘플입니다.

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

UpdateSessionPermissions

특정 NICE DCV 세션에 대한 사용자 권한을 업데이트합니다.

주제

- 요청 파라미터

요청 파라미터

`SessionId`

권한을 변경할 세션의 ID입니다.

유형: 문자열

필수 항목 여부: 예

`Owner`

권한을 변경할 세션의 소유자입니다.

유형: 문자열

필수 항목 여부: 예

`PermissionFile`

권한 파일의 Base64로 인코딩된 내용입니다. 자세한 내용은 NICE DCV 관리자 안내서의 NICE DCV 권한 부여 구성을 참조하세요.

유형: 문자열

필수 항목 여부: 예

응답 파라미터

`SessionId`

세션의 ID입니다.

예

Python

요청

다음 예제에서는 세션 ID가 SessionId1897인 세션의 새로운 권한을 설정합니다.

```
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')])
```

응답

다음은 출력 샘플입니다.

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

NICE DCV 세션 관리자의 릴리스 정보 및 문서 기록

이 페이지에서는 NICE DCV 세션 관리자에 대한 릴리스 정보와 문서 기록을 제공합니다.

주제

NICE DCV 세션 관리자 릴리스 정보

이 섹션에서는 NICE DCV 세션 관리자의 주요 업데이트, 기능 릴리스 정보 및 버그 수정에 대한 개요를 제공합니다. 모든 업데이트는 릴리스 날짜별로 정리되어 있습니다. 사용자로부터 받은 의견을 수렴하기 위해 설명서가 자주 업데이트됩니다.

주제

2023.1— 2023년 11월 9일

빌드 번호	변경 및 버그 수정
• 브로커: 410 • 에이전트: 732 • CLI: 140	• 버그 수정 및 성능 향상

2023.0-15065— 2023년 5월 4일

빌드 번호	변경 및 버그 수정
• 브로커: 392 • 에이전트: 675 • CLI: 132	• ARM 플랫폼에 대한 Red Hat Enterprise Linux 9, Rocky Linux 9, CentOS Stream 9 지원이 추가되었습니다.

2023.0-14852— 2023년 3월 28일

빌드 번호	변경 및 버그 수정
• 브로커: 392 • 에이전트: 642 • CLI: 132	• Red Hat Enterprise Linux 9, Rocky Linux 9, CentOS Stream 9 지원이 추가되었습니다.

2022.2-13907 — 2022년 11월 11일

빌드 번호	변경 및 버그 수정
• 브로커: 382 • 에이전트: 612 • CLI: 123	• `DescribeSessions` 응답에 `Substate` 필드가 추가되었습니다. • 사용 중인 URL에 따라 CLI가 브로커에 연결되지 않는 문제를 해결했습니다.

2022.1-13067— 2022년 6월 29일

빌드 번호	변경 및 버그 수정
• 브로커: 355 • 에이전트: 592 • CLI: 114	• AWS Graviton 인스턴스에서 브로커를 실행하기 위한 지원이 추가되었습니다. • Ubuntu 22.04에 에이전트 및 브로커 지원이 추가되었습니다.

2022.0-11952 — 2022년 2월 23일

빌드 번호	변경 및 버그 수정
• 브로커: 341 • 에이전트: 520 • CLI: 112	• 에이전트에 로그 순환 기능을 추가했습니다. • 브로커에서 Java 홈을 설정하는 구성 파라미터를 추가했습니다. • 브로커의 캐시에서 디스크로의 데이터 플러시를 개선했습니다. • CLI에서 URL 검증을 수정했습니다.

2021.3-11591— 2021년 12월 20일

빌드 번호	새로운 기능
• 브로커: 307 • 에이전트: 453 • CLI: 92	• NICE DCV 연결 게이트웨이 통합에 대한 지원이 추가되었습니다. • Ubuntu 18.04 및 Ubuntu 20.04에 대한 브로커 지원이 추가되었습니다.

2021.2-11445— 2021년 11월 18일

빌드 번호	변경 및 버그 수정
• 브로커: 288 • 에이전트: 413 • CLI: 54	• Windows 도메인이 포함된 로그인 이름을 검증할 때 발생하는 문제를 해결했습니다.

2021.2-11190— 2021년 10월 11일

빌드 번호	변경 및 버그 수정
• 브로커: 254 • 에이전트: 413 • CLI: 54	• Windows 세션을 시작하지 못하게 하는 명령줄 인터페이스 문제를 해결했습니다.

2021.2-11042— 2021년 9월 1일

빌드 번호	새로운 기능	변경 및 버그 수정
• 브로커: 254 • 에이전트: 413 • CLI: 37	• 이제 NICE DCV 세션 관리자가 명령줄 인터페이스(CLI) 지원을 제공합니다. API를 호출하는 대신 CLI에서 NICE DCV 세션을 생성하고 관리할 수 있습니다. • NICE DCV 세션 관리자에 브로커 데이터 지속성이 도입되었습니다. 가용성을 높이기 위해 브로커는 외부 데이터 저장소에 서버 상태 정보를 유지하고 시작 시 데이터를 복원할 수 있습니다.	• 이제 외부 권한 부여 서버를 등록할 때 권한 부여 서버가 JSON 형식의 웹 토큰에 서명하는 데 사용하는 알고리즘을 지정할 수 있습니다. 이번 변경으로 Azure AD를 외부 권한 부여 서버로 사용할 수 있습니다.

2021.1-10557— 2021년 5월 31일

빌드 번호	새로운 기능	변경 및 버그 수정
• 브로커: 214 • 에이전트: 365	• NICE DCV 세션 관리자에 Linux에서 자동 실행 파일에 전달되는 입력 파라미터에 대한 지원이 추가되었습니다. • 이제 서버 속성을 CreateSessions API에 요구 사항으로 전달할 수 있습니다.	• Windows의 자동 실행 파일 관련 문제를 해결했습니다.

2021.0-10242— 2021년 4월 12일

빌드 번호	변경 및 버그 수정

• 브로커: 183
• 에이전트: 318
	• NICE DCV 세션 관리자에 다음과 같은 새로운 API가 도입되었습니다.
	  • OpenServers
	  • CloseServers
	  • DescribeServers
	  • GetSessionScreenshots
	• 또한 다음과 같은 새 구성 파라미터가 도입되었습니다.
	  • 브로커 파라미터: `session-screenshot-max-width` , `session-screenshot-max-height` , `session-screenshot-format` , `create-sessions-queue-max-size` , `create-sessions-queue-max-time-seconds`
	  • 에이전트 파라미터: `agent.autorun_folder` , `max_virtual_sessions` , `max_concurrent_sessions_per_user`
	  에이전트 파라미터: `agent.autorun_folder` , `max_virtual_sessions` , `max_concurrent_sessions_per_user`
	  에이전트 파라미터: `agent.autorun_folder` , `max_virtual_sessions` , `max_concurrent_sessions_per_user`

2020.2-9662— 2020년 12월 4일

빌드 번호	변경 및 버그 수정
• 브로커: 114 • 에이전트: 211	• 브로커가 시작되지 못하게 하는 자동 생성 TLS 인증서 문제를 해결했습니다.

2020.2-9508— 2020년 11월 11일

빌드 번호	변경 및 버그 수정
• 브로커: 78 • 에이전트: 183	• NICE DCV 세션 관리자의 첫 릴리스입니다.

문서 기록

다음 표는 NICE DCV 세션 관리자의 본 릴리스 관련 설명서를 소개합니다.

변경 사항	설명	날짜
NICE DCV 버전 2023.1	NICE DCV 세션 관리자가 NICE DCV 2023.1에 맞게 업데이트되었습니다. 자세한 내용은 2023.1— 2023년 11월 9일 섹션을 참조하세요.	2023년 11월 9일
NICE DCV 버전 2023.0	NICE DCV 세션 관리자가 NICE DCV 2023.0에 맞게 업데이트되었습니다. 자세한 내용은 2023.0-14852— 2023년 3월 28일 섹션을 참조하세요.	2023년 3월 28일
NICE DCV 버전 2022.2	NICE DCV 세션 관리자가 NICE DCV 2022.2에 맞게 업데이트되었습니다. 자세한 내용은 2022.2-13907 — 2022년 11월 11일 섹션을 참조하세요.	2022년 11월 11일
NICE DCV 버전 2022.1	NICE DCV 세션 관리자가 NICE DCV 2022.1에 맞게 업데이트되었습니다. 자세한 내용은 2022.1-13067— 2022년 6월 29일 섹션을 참조하세요.	2022년 6월 29일
NICE DCV 버전 2022.0	NICE DCV 세션 관리자가 NICE DCV 2022.0에 맞게 업데이트되었습니다. 자	2022년 2월 23일

변경 사항	설명	날짜
	세한 내용은 2022.0-11952 — 2022년 2월 23일 섹션을 참조하세요.	
NICE DCV 버전 2021.3	NICE DCV 세션 관리자가 NICE DCV 2021.3에 맞게 업데이트되었습니다. 자세한 내용은 2021.3-11591— 2021년 12월 20일 섹션을 참조하세요.	2021년 12월 20일
NICE DCV 버전 2021.2	NICE DCV 세션 관리자가 NICE DCV 2021.2에 맞게 업데이트되었습니다. 자세한 내용은 2021.2-11042— 2021년 9월 1일 섹션을 참조하세요.	2021년 9월 1일
NICE DCV 버전 2021.1	NICE DCV 세션 관리자가 NICE DCV 2021.1에 맞게 업데이트되었습니다. 자세한 내용은 2021.1-10557— 2021년 5월 31일 섹션을 참조하세요.	2021년 5월 31일
NICE DCV 버전 2021.0	NICE DCV 세션 관리자가 NICE DCV 2021.0에 맞게 업데이트되었습니다. 자세한 내용은 2021.0-10242— 2021년 4월 12일 섹션을 참조하세요.	2021년 4월 12일
NICE DCV 세션 관리자의 첫 릴리스	이 내용의 첫 번째 발행입니다.	2020년 11월 11일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.