aws

Entwicklerhandbuch

NICEDCVSitzungsmanager

NICEDCVSitzungsmanager: Entwicklerhandbuch

Table of Contents

Was ist Session Manager? 1
Funktionsweise von Session Manager 1
Funktionen 3
Erste Schritte mit Session Manager API 5
Schritt 1: Generieren Sie Ihren Client API 5
Schritt 2: Registrieren Sie Ihren Kunden API 6
Schritt 3: Besorgen Sie sich ein Zugriffstoken und stellen Sie eine API Anfrage 7
Session API Manager-Referenz 10
CloseServers 10
Anforderungsparameter 7
Antwortparameter 11
Beispiel 12
CreateSessions 13
Anforderungsparameter 7
Antwortparameter 11
Beispiel 12
DescribeServers 21
Anforderungsparameter 7
Antwortparameter 11
Beispiel 12
DescribeSessions 32
Anforderungsparameter 7
Antwortparameter 11
Beispiel 12
DeleteSessions 39
Anforderungsparameter 7
Antwortparameter 11
Beispiel 12
GetSessionConnectionData 42
Anforderungsparameter 7
Antwortparameter 11
Zusätzliche Informationen 45
Beispiel 12
GetSessionScreenshots 48

Anforderungsparameter 7
Antwortparameter 11
Beispiel 12
OpenServers 51
Anforderungsparameter 7
Antwortparameter 11
Beispiel 12
UpdateSessionPermissions 53
Anforderungsparameter 7
Antwortparameter 11
Beispiel 12
Versionshinweise und Dokumentverlauf 56
Versionshinweise 56
2023.1-17652 — 1. August 2024 57
2023.1-16388 — 26. Juni 2024 57
2023.1 — 9. November 2023 57
2023.0-15065 — 4. Mai 2023 57
2023.0-14852 — 28. März 2023 58
2022.2-13907 — 11. November 2022 58
2022.1-13067 — 29. Juni 2022 58
2022.0-11952 — 23. Februar 2022 59
2021.3-11591 — 20. Dezember 2021 59
2021.2-11445 — 18. November 2021 59
2021.2-11190 — 11. Oktober 2021 60
2021.2-11042 — 01. September 2021 60
2021.1-10557 — 31. Mai 2021 61
2021.0-10242 — 12. April 2021 61
2020.2-9662 — 04. Dezember 2020 62
..... 62
Dokumentverlauf 62
..... lxv

Was ist NICE DCV Session Manager?

NICE DCV Session Manager besteht aus installierbaren Softwarepaketen (ein Agent und ein Broker) und einer Anwendungsprogrammierschnittstelle (API), die es Entwicklern und unabhängigen Softwareanbietern (ISVs) leicht macht, Frontend-Anwendungen zu erstellen, die programmgesteuert den Lebenszyklus von NICE-DCV-Sitzungen auf einer Flotte von NICE-DCV-Servern erstellen und verwalten.

In diesem Handbuch wird erklärt, wie Sie die Session Manager-APIs verwenden, um den Lebenszyklus von NICE-DCV-Sitzungen zu verwalten. Weitere Informationen zur Installation und Konfiguration des Session Manager Brokers und der Agents finden Sie im NICE DCV Session Manager Administrator Guide.

Voraussetzungen

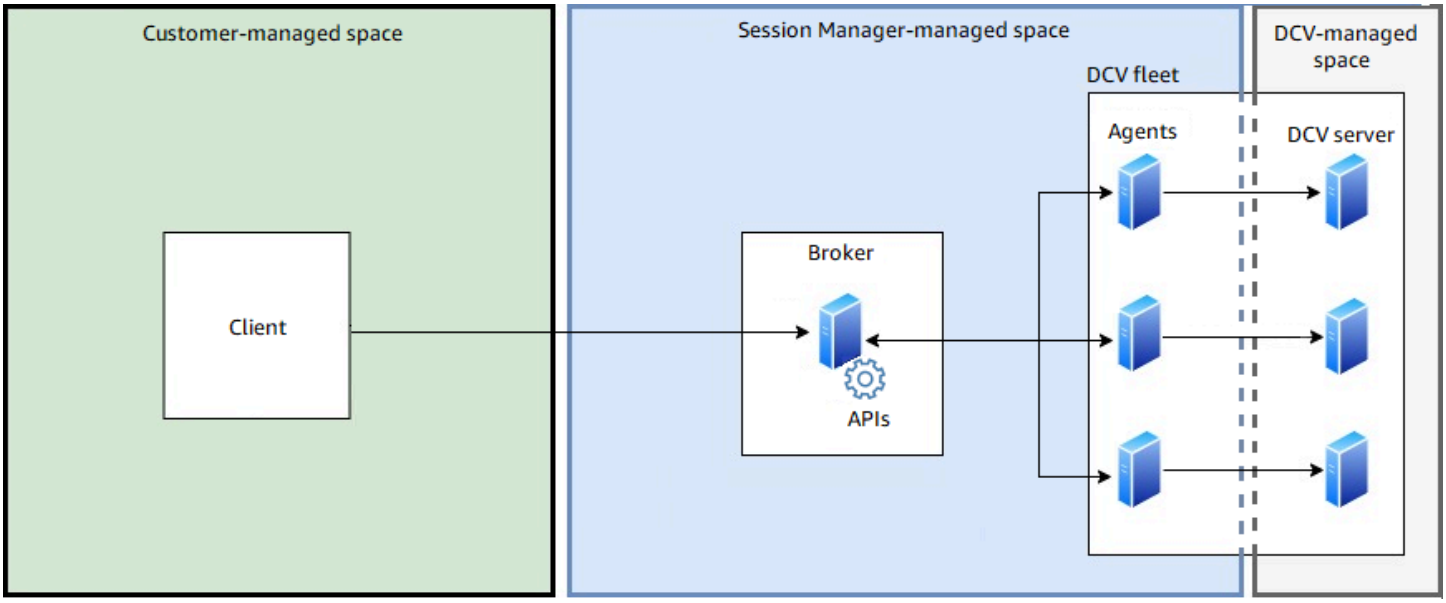
Bevor Sie mit der Arbeit mit den Session Manager-APIs beginnen, stellen Sie sicher, dass Sie mit NICE DCV- und NICE DCV-Sitzungen vertraut sind. Weitere Informationen finden Sie im NICE DCV Administratorhandbuch.

Themen

- Funktionsweise von Session Manager
- Funktionen

Funktionsweise von Session Manager

Das folgende Diagramm zeigt die folgenden Komponenten von Session Manager.



Broker

Der Broker ist ein Webserver, der die Session Manager-APIs hostet und verfügbar macht. Es empfängt und verarbeitet API-Anfragen zur Verwaltung von NICE-DCV-Sitzungen vom Client und leitet die Anweisungen dann an die entsprechenden Agenten weiter. Der Broker muss auf einem Host installiert sein, der von Ihren NICE-DCV-Servern getrennt ist, aber er muss für den Client zugänglich sein und auf die Agents zugreifen können.

Kundendienstmitarbeiter

Der Agent ist auf jedem NICE-DCV-Server in der Flotte installiert. Die Agents erhalten Anweisungen vom Broker und führen sie auf ihren jeweiligen NICE-DCV-Servern aus. Die Agents überwachen auch den Status der NICE-DCV-Server und senden regelmäßige Statusaktualisierungen an den Broker zurück.

APIs

Session Manager stellt eine Reihe von REST-Anwendungsprogrammierschnittstellen (APIs) zur Verfügung, mit denen NICE-DCV-Sitzungen auf einer Flotte von NICE-DCV-Servern verwaltet werden können. Die APIs werden auf dem Broker gehostet und von diesem verfügbar gemacht. Entwickler können benutzerdefinierte Sitzungsverwaltungsclients erstellen, die die APIs aufrufen.

Client

Der Client ist die Frontend-Anwendung oder das Frontend-Portal, das Sie entwickeln, um die vom Broker bereitgestellten Session Manager-APIs aufzurufen. Endbenutzer verwenden den Client, um die auf den NICE-DCV-Servern der Flotte gehosteten Sitzungen zu verwalten.

Zugriffstoken

Um eine API-Anfrage zu stellen, müssen Sie ein Zugriffstoken angeben. Token können über registrierte Client-APIs vom Broker oder einem externen Autorisierungsserver angefordert werden. Um das Token anzufordern und darauf zuzugreifen, muss die Client-API gültige Anmeldeinformationen angeben.

Client-API

Die Client-API wird mithilfe von Swagger Codegen aus der Session Manager-API-Definitionsdatei generiert. Die Client-API wird verwendet, um API-Anfragen zu stellen.

NICE DCV-Sitzung

Sie müssen eine NICE-DCV-Sitzung auf Ihrem NICE-DCV-Server erstellen, zu der Ihre Clients eine Verbindung herstellen können. Clients können nur dann eine Verbindung zu einem NICE-DCV-Server herstellen, wenn eine aktive Sitzung besteht. NICE DCV unterstützt Konsolen- und virtuelle Sitzungen. Sie verwenden die Session Manager-APIs, um den Lebenszyklus von NICE-DCV-Sitzungen zu verwalten. NICE-DCV-Sitzungen können sich in einem der folgenden Zustände befinden:

- `CREATING`— Der Broker ist dabei, die Sitzung zu erstellen.
- `READY`— Die Sitzung ist bereit, Client-Verbindungen zu akzeptieren.
- `DELETING`—Die -Anfrage wird die folgenden Zustände aufweisen.
- `DELETED`—Die -Anfrage wurde die folgenden Zustände aufweisen.
- `UNKNOWN`— der Status der Sitzung kann nicht ermittelt werden. Der Broker und der Agent können möglicherweise nicht kommunizieren.

Funktionen

Der DCV Session Manager bietet die folgenden Funktionen:

- Stellt NICE-DCV-Sitzungsinformationen bereit — Hier erhalten Sie Informationen über die Sitzungen, die auf mehreren NICE-DCV-Servern laufen.
- Verwalten Sie den Lebenszyklus für mehrere NICE-DCV-Sitzungen — erstellen oder löschen Sie mehrere Sitzungen für mehrere Benutzer auf mehreren NICE-DCV-Servern mit einer API-Anfrage.
- Unterstützt Tags — verwenden Sie benutzerdefinierte Tags, um beim Erstellen von Sitzungen eine Gruppe von NICE-DCV-Servern anzusprechen.

- Verwaltet Berechtigungen für mehrere NICE-DCV-Sitzungen — ändern Sie Benutzerberechtigungen für mehrere Sitzungen mit einer API-Anfrage.
- Stellt Verbindungsinformationen bereit — ruft Client-Verbindungsinformationen für NICE-DCV-Sitzungen ab.
- Unterstützung für die Cloud und vor Ort — verwenden Sie Session Manager vor OrtAWS, vor Ort oder mit alternativen cloudbasierten Servern.

Erste Schritte mit Session Manager API

Der NICE DCV Session Manager API bietet eine automatisierte Oberfläche für die Verwaltung von Remote-Desktop-Sitzungen. Auf diese API Weise können Entwickler DCV Sitzungen programmgesteuert erstellen, auflisten, starten, beenden und anderweitig steuern. Dies ermöglicht die Integration von NICE DCV Funktionen in benutzerdefinierte Anwendungen und Workflows. Auf diese API Weise können Unternehmen die Verwaltung von Remote-Visualisierungs-Workloads optimieren und viele allgemeine Aufgaben automatisieren.

Bevor Sie mit dem Aufrufen von beginnen können NICE DCVAPI, benötigen Sie ein Zugriffstoken, das Ihre Anwendung authentifiziert und sie für den Zugriff auf die erforderlichen Ressourcen autorisiert. Der NICE DCV API verwendet OAuth 2.0 für die Authentifizierung, sodass Sie Ihre Anwendung registrieren und die erforderlichen Anmeldeinformationen abrufen müssen. Sobald Sie Ihr Zugriffstoken haben, können Sie damit beginnen, Anfragen an die NICE DCV API Endgeräte zu senden, um mit der Datenverarbeitung zu beginnen.

Themen

- Schritt 1: Generieren Sie Ihren Client API
- Schritt 2: Registrieren Sie Ihren Kunden API
- Schritt 3: Besorgen Sie sich ein Zugriffstoken und stellen Sie eine API Anfrage

Schritt 1: Generieren Sie Ihren Client API

Die Session Manager APIs sind in einer einzigen YAML Datei definiert. Sie APIs basieren auf der Open API3 4.0-Spezifikation, die eine standardmäßige, sprachunabhängige Schnittstelle zu definiert. RESTful APIs Weitere Informationen finden Sie unter Open Specification. API

Mithilfe der YAML Datei können Sie einen API Client in einer der unterstützten Sprachen generieren. Dazu müssen Sie Swagger Codegen 3.0 oder höher verwenden. Weitere Informationen zu den unterstützten Sprachen finden Sie im Swagger-Codegen-Repo.

APIUm den Client zu generieren

1. Laden Sie die Session API YAML Manager-Datei vom Session Manager Broker herunter. Die YAML Datei ist im Folgenden verfügbarURL.

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

2. Installieren Sie Swagger Codegen.

   - macOS

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

   - Andere Plattformen

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

     ```
     $ cd swagger-codegen
     ```

3. Generieren Sie den API Client.

   - macOS

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

   - Andere Plattformen

     ```
     $ 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
     ```

Schritt 2: Registrieren Sie Ihren Kunden API

APIAnfragen verwenden ein Zugriffstoken, um Ihre Anmeldeinformationen zu überprüfen. Diese Anmeldeinformationen basieren auf einer Client-ID und einem Client-Passwort, die generiert werden, wenn Ihr Kunde beim Broker registriert wird.

Um auf dieses Token zugreifen zu können, müssen Sie sich beim Broker registrieren. Wird verwendet register-api-client, um den Client zu registrierenAPI.

Wenn Sie keine Kunden-ID und kein Kundenkennwort für Ihren Kunden haben, müssen Sie diese von Ihrem Broker-Administrator anfordern.

Schritt 3: Besorgen Sie sich ein Zugriffstoken und stellen Sie eine API Anfrage

In diesem Beispiel werden die Schritte zur Einrichtung Ihres Zugriffstokens beschrieben. Anschließend wird gezeigt, wie Sie eine einfache API Anfrage stellen. Auf diese Weise erhalten Sie das grundlegende Wissen, um mit der Entwicklung fortgeschrittenerer Anwendungen zu beginnen, die auf dem NICE DCV API basieren.

In diesem Beispiel zeigen wir Ihnen, wie Sie dies mit dem `DescribeSessions` API tun können.

Example

Zuerst importieren wir die für die Anwendung benötigten Modelle.

Dann deklarieren wir Variablen für die Client-ID (`__CLIENT_ID`), das Client-Passwort (`__CLIENT_SECRET`) und den BrokerURL, einschließlich der Portnummer (`__PROTOCOL_HOST_PORT`).

Als Nächstes erstellen wir eine Funktion namens`build_client_credentials`, die die Client-Anmeldeinformationen generiert. Um die Client-Anmeldeinformationen zu generieren, müssen Sie zuerst die Client-ID und das Client-Passwort verketten und die Werte durch einen Doppelpunkt (*`client_ID`*`:`*`client_password`*) trennen. Anschließend müssen Sie die gesamte Zeichenfolge mit Base64 kodieren.

```
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')
```

Da wir nun unsere Client-Anmeldeinformationen haben, können wir sie verwenden, um ein Zugriffstoken vom Broker anzufordern. Dazu erstellen wir eine Funktion namens`get_access_token`. Sie müssen ein POST on `https://Broker_IP:8443/oauth2/token?grant_type=client_credentials` aufrufen und einen Autorisierungsheader angeben, der die BASIC-kodierten Client-Anmeldeinformationen und den Inhaltstyp enthält. `application/x-www-form-urlencoded`

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

Jetzt erstellen wir die Funktionen, die für die Instanziierung eines Clients erforderlich sind. API Um einen Client zu instanziierenAPI, müssen Sie die Client-Konfiguration und die Header angeben, die für Anfragen verwendet werden sollen. Die `get_client_configuration` Funktion erstellt ein Konfigurationsobjekt, das die IP-Adresse und den Port des Brokers sowie den Pfad zum selbstsignierten Zertifikat des Brokers enthält, das Sie vom Broker-Administrator erhalten haben sollten. Die `set_request_headers` Funktion erstellt ein Anforderungsheader-Objekt, das die Client-Anmeldeinformationen und das Zugriffstoken enthält.

```
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
```

Schließlich erstellen wir eine Hauptmethode, die den aufruft `DescribeSessions`API. Weitere Informationen finden Sie unter DescribeSessions.

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

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

def main():
    describe_sessions(
        session_ids=['SessionId1895', 'SessionId1897'],
        owner='an owner 1890',
        tags=[{'Key': 'ram', 'Value': '4gb'}])
```

Session API Manager-Referenz

Diese Referenz enthält Einzelheiten zu den verfügbaren API Aktionen, erforderlichen Parametern und Antwortformaten, damit Sie den Session Manager effektiv API in Ihren eigenen Systemen nutzen können. Mit dem Session Manager API können Sie interaktive Sitzungen starten, beenden und Details zu diesen abrufen. Auf diese Weise können Sie Funktionen automatisieren und in Ihre Anwendungen und Workflows integrieren.

Themen

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

CloseServers

Schließt einen oder mehrere NICE DCV Server. Wenn Sie einen NICE DCV Server schließen, ist er für die NICE DCV Sitzungsplatzierung nicht verfügbar. Sie können keine NICE DCV Sitzungen auf geschlossenen Servern erstellen. Durch das Schließen eines Servers wird sichergestellt, dass keine Sitzungen auf dem Server ausgeführt werden und dass Benutzer keine neuen Sitzungen auf dem Server erstellen können.

Themen

- Anforderungsparameter
- Antwortparameter
- Beispiel

Anforderungsparameter

`ServerId`

Die ID des Servers, der geschlossen werden soll.

Typ: Zeichenfolge

Erforderlich: Ja

`Force`

Erzwingt den Schließvorgang. Wenn Sie dies angeben`true`, wird der Server geschlossen, auch wenn er laufende Sitzungen hat. Die Sitzungen werden weiterhin ausgeführt.

Typ: Boolesch

Erforderlich: Nein

Antwortparameter

`RequestId`

Die eindeutige ID der Anfrage.

`SuccessfulList`

Informationen über die NICE DCV Server, die erfolgreich geschlossen wurden. Diese Datenstruktur umfasst den folgenden verschachtelten Antwortparameter:

`ServerId`

Die ID des Servers, der erfolgreich geschlossen wurde.

`UnsuccessfulList`

Informationen zu den NICE DCV Servern, die nicht geschlossen werden konnten. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`CloseServerRequestData`

Informationen über die ursprüngliche Anfrage, die fehlgeschlagen ist. Diese Datenstruktur umfasst den folgenden verschachtelten Antwortparameter:

`ServerId`

Die ID des NICE DCV Servers, der nicht geschlossen werden konnte.

`Force`

Der angeforderte Force-Parameter.

`FailureCode`

Der Code des Fehlers.

`FailureReason`

Der Grund für den Fehlschlag.

Beispiel

Python

Anforderung

Das folgende Beispiel schließt zwei NICE DCV Server (`serverId1`und`serverId2`). Der Server `serverId2` ist nicht vorhanden und führt zu einem Ausfall.

```
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"])
```

Antwort

Im Folgenden finden Sie ein Beispiel für die Ausgabe.

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

CreateSessions

Erstellt eine neue NICE DCV Sitzung mit den angegebenen Details.

APIAktionen

- Anforderungsparameter
- Antwortparameter
- Beispiel

Anforderungsparameter

`Name`

Der Name der Sitzung.

Typ: Zeichenfolge

Erforderlich: Ja

`Owner`

Der Name des Sitzungsbesitzers. Dies muss der Name eines vorhandenen Benutzers auf dem NICE DCV Zielserver sein.

Typ: Zeichenfolge

Erforderlich: Ja

`Type`

Der Sitzungstyp Weitere Informationen zu den Sitzungstypen finden Sie unter Einführung in NICE DCV Sitzungen im NICEDCVAdministratorhandbuch.

Gültige Werte: CONSOLE | VIRTUAL

Typ: Zeichenfolge

Erforderlich: Ja

`InitFile`

Wird bei virtuellen Sitzungen auf NICE DCV Linux-Servern unterstützt. Es wird bei Konsolensitzungen auf Windows- und NICE DCV Linux-Servern nicht unterstützt. Der Pfad zum benutzerdefinierten Skript auf dem NICE DCV Server, das für die Initialisierung der Sitzung ausgeführt werden soll, wenn sie erstellt wird. Der Dateipfad ist relativ zu dem für den `agent.init_folder` Agent-Konfigurationsparameter angegebenen Init-Verzeichnis. Wenn sich die Datei im angegebenen Init-Verzeichnis befindet, geben Sie nur den Dateinamen an. Wenn sich die Datei nicht im angegebenen Init-Verzeichnis befindet, geben Sie den relativen Pfad an. Weitere Informationen finden Sie unter Agent-Konfigurationsdatei im NICEDCVSession Manager-Administratorhandbuch.

Typ: Zeichenfolge

Erforderlich: Nein

`MaxConcurrents`

Die maximale Anzahl gleichzeitiger NICE DCV Clients.

Typ: Ganzzahl

Erforderlich: Nein

`DcvGlEnabled`

Gibt an, ob die virtuelle Sitzung für die Verwendung von hardwarebasiertem OpenGL konfiguriert ist. Wird nur bei virtuellen Sitzungen unterstützt. Dieser Parameter wird auf NICE DCV Windows-Servern nicht unterstützt.

Zulässige Werte: true | false

Typ: Boolesch

Erforderlich: Nein

`PermissionsFile`

Der Base64-kodierte Inhalt der Berechtigungsdatei. Standardmäßig die Server-Standardwerte, falls kein Wert angegeben ist. Weitere Informationen finden Sie unter Konfiguration der NICE DCV Autorisierung im Administratorhandbuch. NICE DCV

Typ: Zeichenfolge

Erforderlich: Nein

`EnqueueRequest`

Gibt an, ob die Anfrage in die Warteschlange gestellt werden soll, wenn sie nicht sofort erfüllt werden kann.

Typ: Boolesch

Standard: false

Erforderlich: Nein

`AutorunFile`

Wird bei Konsolensitzungen auf NICE DCV Windows-Servern und virtuellen Sitzungen auf NICE DCV Linux-Servern unterstützt. Es wird bei Konsolensitzungen auf NICE DCV Linux-Servern nicht unterstützt.

Der Pfad zu einer Datei auf dem Hostserver, die innerhalb der Sitzung ausgeführt werden soll. Der Dateipfad ist relativ zu dem Autorun-Verzeichnis, das für den `agent.autorun_folder` Agent-Konfigurationsparameter angegeben wurde. Wenn sich die Datei im angegebenen Autorun-Verzeichnis befindet, geben Sie nur den Dateinamen an. Wenn sich die Datei nicht im angegebenen Autorun-Verzeichnis befindet, geben Sie den relativen Pfad an. Weitere

Informationen finden Sie in der Agent-Konfigurationsdatei im NICEDCVSession Manager-Administratorhandbuch.

Die Datei wird im Namen des angegebenen Besitzers ausgeführt. Der angegebene Besitzer muss berechtigt sein, die Datei auf dem Server auszuführen. Auf NICE DCV Windows-Servern wird die Datei ausgeführt, wenn sich der Besitzer bei der Sitzung anmeldet. Auf NICE DCV Linux-Servern wird die Datei ausgeführt, wenn die Sitzung erstellt wird.

Typ: Zeichenfolge

Erforderlich: Nein

`AutorunFileArguments`

Wird bei virtuellen Sitzungen auf NICE DCV Linux-Servern unterstützt. Es wird in Konsolensitzungen auf Windows- und NICE DCV Linux-Servern nicht unterstützt. Befehlszeilenargumente, die AutorunFilebei der Ausführung innerhalb der Sitzung übergeben werden. Argumente werden in der Reihenfolge übergeben, in der sie im angegebenen Array erscheinen. Die maximal zulässige Anzahl von Argumenten und die maximal zulässige Länge jedes Arguments können konfiguriert werden. Weitere Informationen finden Sie in der Broker-Konfigurationsdatei im NICEDCVSession Manager-Administratorhandbuch.

Typ: Zeichenfolgen-Array

Erforderlich: Nein

`DisableRetryOnFailure`

Gibt an, ob die Anfrage zum Erstellen einer Sitzung nicht erneut versucht werden soll, wenn sie auf einem NICE DCV Host aus irgendeinem Grund fehlschlägt. Weitere Informationen zum Mechanismus zum erneuten Versuch beim Erstellen einer Sitzung finden Sie in der Broker-Konfigurationsdatei im NICEDCVSession Manager-Administratorhandbuch.

Typ: Boolesch

Standard: false

Erforderlich: Nein

`Requirements`

Die Anforderungen, die der Server erfüllen muss, um die Sitzung durchführen zu können. Die Anforderungen können Server-Tags und/oder Servereigenschaften beinhalten. Sowohl Server-Tags als auch Servereigenschaften werden durch Aufrufen von abgerufen DescribeServersAPI.

Bedingungsausdrücke für Anforderungen:

- *a* != *b* wahr, wenn *a* ist nicht gleich *b*
- *a* = *b* wahr wenn *a* ist gleich *b*
- *a* > *b* wahr wenn *a* ist größer als *b*
- *a* >= *b* wahr wenn *a* ist größer als oder gleich *b*
- *a* < *b* wahr wenn *a* ist kleiner als *b*
- *a* <= *b* wahr wenn *a* ist kleiner als oder gleich *b*
- *a* = *b* wahr wenn *a* enthält die Zeichenfolge *b*

Anforderungen boolesche Operatoren:

- *a* und *b* wahr wenn *a* and *b* sind wahr
- *a* oder *b* wahr wenn *a* or *b* sind wahr
- nicht *a* wahr wenn *a* ist falsch

Den Tag-Schlüsseln muss ein Präfix vorangestellt werden`tag:`, den Servereigenschaften muss ein Präfix vorangestellt werden. `server:` Die Anforderungsausdrücke unterstützen Klammern. `()`

Beispiele für Anforderungen:

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

Numerische Werte können in exponentieller Notation angegeben werden, zum Beispiel:`"server:Host.Memory.TotalBytes > 1024E6"`.

Die unterstützten Servereigenschaften sind:

- `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`

Typ: Zeichenfolge

Erforderlich: Nein

`StorageRoot`

Gibt den Pfad zu dem Ordner an, der als Speicher für die Sitzung verwendet wird. Weitere Informationen zum NICE DCV Sitzungsspeicher finden Sie unter Aktivieren des Sitzungsspeichers im NICEDCVAdministratorhandbuch.

Typ: Zeichenfolge

Erforderlich: Nein

Antwortparameter

`Id`

Die eindeutige ID der Sitzung.

`Name`

Der Sitzungsname

`Owner`

Der Besitzer der Sitzung.

`Type`

Die Art der Sitzung.

`State`

Der Status der Sitzung. Wenn die Anfrage erfolgreich abgeschlossen wurde, wechselt die Sitzung in den `CREATING` Status.

`Substate`

Der Unterstatus der Sitzung. Wenn die Anforderung erfolgreich abgeschlossen wurde, wechselt der Unterstatus in den `SESSION_PLACING` Unterstatus.

Beispiel

Python

Anforderung

Im folgenden Beispiel werden drei Sitzungen erstellt.

```
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),
])
```

Antwort

Das Folgende ist die Beispielausgabe.

```
{
              "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

Beschreibt einen oder mehrere NICE DCV Server.

Themen

- Anforderungsparameter
- Antwortparameter
- Beispiel

Anforderungsparameter

`ServerIds`

Der NICE DCV Server, IDs der beschrieben werden soll. Wenn keine angegeben IDs sind, werden alle Server in einer paginierten Ausgabe zurückgegeben.

Typ: Zeichenfolgen-Array

Erforderlich: Nein

`NextToken`

Das Token, das zum Abrufen der nächsten Ergebnisseite verwendet werden soll.

Typ: Zeichenfolge

Erforderlich: Nein

`MaxResults`

Die maximale Anzahl von Ergebnissen, die von der Anforderung in einer paginierten Ausgabe zurückgegeben werden sollen. Wenn dieser Parameter verwendet wird, gibt die Anforderung nur die angegebene Anzahl von Ergebnissen auf einer einzelnen Seite zusammen mit einem `NextToken` Antwortelement zurück. Die verbleibenden Ergebnisse der ersten Anfrage können angezeigt werden, indem eine weitere Anfrage mit dem zurückgegebenen `NextToken` Wert gesendet wird.

Gültiger Bereich: 1–1 000

Standard: 1000

Typ: Ganzzahl

Erforderlich: Nein

Antwortparameter

`RequestId`

Die eindeutige ID der Anfrage.

`Servers`

Informationen über die NICE DCV Server. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`Id`

Die eindeutige ID des NICE DCV Servers.

`Ip`

Die IP-Adresse des NICE DCV Servers.

`Hostname`

Der Hostname des NICE DCV Servers.

`Endpoints`

Informationen zu den NICE DCV Serverendpunkten. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`IpAddress`

Die IP-Adresse des Serverendpunkts.

`Port`

Der Port des Serverendpunkts.

`Protocol`

Das vom Serverendpunkt verwendete Protokoll. Mögliche Werte sind:

- `HTTP`— Der Endpunkt verwendet das Protokoll WebSocket (TCP).
- `QUIC`— Der Endpunkt verwendet das QUIC (UDP) -Protokoll.

`WebUrlPath`

Der URL Webpfad des Serverendpunkts. Nur für das HTTP Protokoll verfügbar.

`Version`

Die Version des NICE DCV Servers.

`SessionManagerAgentVersion`

Die Version des Session Manager Agents, die auf dem NICE DCV Server ausgeführt wird.

`Availability`

Die Verfügbarkeit des NICE DCV Servers. Mögliche Werte sind:

- `AVAILABLE`— Der Server ist verfügbar und bereit für die Sitzungsplatzierung.
- `UNAVAILABLE`— Der Server ist nicht verfügbar und kann die Sitzungsplatzierung nicht akzeptieren.

`UnavailabilityReason`

Der Grund für die Nichtverfügbarkeit des NICE DCV Servers. Mögliche Werte sind:

- `SERVER_FULL`— Der NICE DCV Server hat die maximale Anzahl gleichzeitiger Sitzungen erreicht, die er ausführen kann.

- `SERVER_CLOSED`— Der NICE DCV Server wurde mithilfe von nicht verfügbar gemacht. CloseServerAPI
- `UNREACHABLE_AGENT`— Der Session Manager Broker kann nicht mit dem Session Manager Agent auf dem NICE DCV Server kommunizieren.
- `UNHEALTHY_DCV_SERVER`— Der Session Manager Agent kann nicht mit dem NICE DCV Server kommunizieren.
- `EXISTING_LOGGED_IN_USER`— (Nur NICE DCV Windows-Server) Ein Benutzer ist derzeit am NICE DCV Server angemeldet mitRDP.
- `UNKNOWN`— Der Session Manager Broker kann den Grund nicht ermitteln.

`ConsoleSessionCount`

Die Anzahl der Konsolensitzungen auf dem NICE DCV Server.

`VirtualSessionCount`

Die Anzahl der virtuellen Sitzungen auf dem NICE DCV Server.

`Host`

Informationen über den Hostserver, auf dem der NICE DCV Server läuft. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`Os`

Informationen zum Betriebssystem des Hostservers. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`Family`

Die Betriebssystemfamilie. Mögliche Werte sind:

- `windows`— Auf dem Host-Server wird ein Windows-Betriebssystem ausgeführt.
- `linux`— Auf dem Host-Server wird ein Linux-Betriebssystem ausgeführt.

`Name`

Der Name des Betriebssystems.

`Version`

Die Version des Betriebssystems.

`KernelVersion`

(Nur Linux) Die Kernelversion des Betriebssystems.

`BuildNumber`

(Nur Windows) Die Buildnummer des Betriebssystems.

`Memory`

Informationen über den Arbeitsspeicher des Hostservers. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`TotalBytes`

Der Gesamtspeicher auf dem Hostserver in Byte.

`UsedBytes`

Der verwendete Speicher auf dem Hostserver in Byte.

`Swap`

Informationen über die Swap-Datei des Hostservers. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`TotalBytes`

Die Gesamtgröße der Swap-Datei auf dem Hostserver in Byte.

`UsedBytes`

Die verwendete Größe der Swap-Datei in Byte auf dem Hostserver.

`Aws`

Nur für NICE DCV Server, die auf einer EC2 Amazon-Instance laufen. AWS-spezifische Informationen. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`Region`

Das Tool AWS Region der EC2 Amazon-Instance.

`Ec2InstanceType`

Der Typ der EC2 Amazon-Instance.

`Ec2InstanceId`

Die ID der EC2 Amazon-Instance.

`Ec2ImageId`

Die ID des EC2 Amazon-Bildes.

CpuInfo

Informationen über die des HostserversCPUs. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

Vendor

Der Anbieter des HostserversCPU.

ModelName

Der Modellname des HostserversCPU.

Architecture

Die Architektur des HostserversCPU.

NumberOfCpus

Die Nummer von CPUs auf dem Hostserver.

PhysicalCorePerCpu

Die Anzahl der CPU Kerne proCPU.

CpuLoadAverage

Informationen über die CPU Auslastung des Hostservers. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

OneMinute

Die durchschnittliche CPU Auslastung im letzten Zeitraum von 1 Minute.

FiveMinutes

Die durchschnittliche CPU Auslastung in den letzten 5 Minuten.

FifteenMinutes

Die durchschnittliche CPU Auslastung in den letzten 15 Minuten.

Gpus

Informationen über die des HostserversGPUs. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

Vendor

Der Anbieter des HostserversGPU.

ModelName

Der Modellname des HostserversGPU.

LoggedInUsers

Die Benutzer, die derzeit am Hostserver angemeldet sind. Diese Datenstruktur umfasst den folgenden verschachtelten Antwortparameter:

Username

Der Benutzername des angemeldeten Benutzers.

Tags

Die dem Server zugewiesenen Tags. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

Key

Der Tag-Schlüssel.

Value

Der Tag-Wert.

Beispiel

Python

Anforderung

Das folgende Beispiel beschreibt alle verfügbaren NICE DCV Server. Die Ergebnisse sind paginiert, sodass zwei Ergebnisse pro Seite angezeigt werden.

```
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)
```

Antwort

Im Folgenden finden Sie ein Beispiel für die Ausgabe.

```
{
    "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

Beschreibt eine oder mehrere NICE DCV Sitzungen.

Themen

- Anforderungsparameter
- Antwortparameter
- Beispiel

Anforderungsparameter

`SessionIds`

Die IDs zu beschreibenden Sitzungen.

Typ: Zeichenfolge

Erforderlich: Nein

`NextToken`

Das Token, das zum Abrufen der nächsten Ergebnisseite verwendet werden soll.

Typ: Zeichenfolge

Erforderlich: Nein

`Filters`

Zusätzliche Filter, die auf die Anfrage angewendet werden sollen. Zu den unterstützten Filtern gehören:

- tag:key — Die der Sitzung zugewiesenen Tags.
- Besitzer — Der Besitzer der Sitzung.

Typ: Zeichenfolge

Erforderlich: Nein

Antwortparameter

`Id`

Die eindeutige ID der Sitzung.

`Name`

Der Name der Sitzung.

`Owner`

Der Besitzer der Sitzung.

`Server`

Informationen über den Server, auf dem die Sitzung läuft. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`Ip`

Die IP-Adresse des NICE DCV Serverhosts.

`Hostname`

Der Hostname des NICE DCV Serverhosts.

`Port`

Der Port, über den der NICE DCV Server mit NICE DCV den Clients kommuniziert.

Endpoints

Informationen zu den NICE DCV Serverendpunkten. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

IpAddress

Die IP-Adresse des Serverendpunkts.

Port

Der Port des Serverendpunkts.

Protocol

Das vom Serverendpunkt verwendete Protokoll. Mögliche Werte sind:

- `HTTP`— Der Endpunkt verwendet das Protokoll WebSocket (TCP).
- `QUIC`— Der Endpunkt verwendet das QUIC (UDP) -Protokoll.

WebUrlPath

Der URL Webpfad des Serverendpunkts. Nur für das HTTP Protokoll verfügbar.

Tags

Die dem Server zugewiesenen Tags. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

Key

Der Tag-Schlüssel.

Value

Der Tag-Wert.

Type

Die Art der Sitzung.

State

Der aktuelle Status der Sitzung. Die möglichen Werte sind:

- `CREATING`- Der Broker ist dabei, die Sitzung zu erstellen.
- `READY`- Die Sitzung ist bereit, Client-Verbindungen anzunehmen.
- `DELETING`- Die Sitzung wird gelöscht.
- `DELETED`- Die Sitzung wurde gelöscht.
- `UNKNOWN`- Der Status der Sitzung konnte nicht ermittelt werden. Der Broker und der Agent können möglicherweise nicht kommunizieren.

`Substate`

Der aktuelle Unterstatus der Sitzung. Die möglichen Werte sind:

- `SESSION_PLACING`- Die Sitzung wartet darauf, auf einem verfügbaren DCV Server platziert zu werden.
- `PENDING_PREPARATION`- Die Sitzung wurde erstellt, ist aber nicht nutzbar; sie ist mit einem DCV Server verknüpft.

`CreationTime`

Datum und Uhrzeit der Erstellung der Sitzung.

`LastDisconnectionTime`

Datum und Uhrzeit der letzten Verbindungsunterbrechung des Clients.

`NumOfConnections`

Die Anzahl der aktiven Client-Verbindungen.

`StorageRoot`

Gibt den Pfad zu dem Ordner an, der als Speicher für die Sitzung verwendet wird. Weitere Informationen zum NICE DCV Sitzungsspeicher finden Sie unter Aktivieren des Sitzungsspeichers im NICEDCVAdministratorhandbuch.

Typ: Zeichenfolge

Erforderlich: Nein

Beispiel

Python

Anforderung

Im folgenden Beispiel werden Sitzungen beschrieben, die Eigentum von sind `user1` und das Tag von haben`os=windows`.

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

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

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

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

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

Antwort

Das Folgende ist die Beispielausgabe.


```
{
    "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

Löscht die angegebene NICE DCV Sitzung und entfernt sie aus dem Cache des Brokers.

Themen

- Anforderungsparameter
- Antwortparameter
- Beispiel

Anforderungsparameter

`SessionId`

Die ID der zu löschenden Sitzung.

Typ: Zeichenfolge

Erforderlich: Ja

`Owner`

Der Besitzer der zu löschenden Sitzung.

Typ: Zeichenfolge

Erforderlich: Ja

`Force`

Löscht eine Sitzung aus dem Cache des Brokers und versucht, sie vom NICE DCV Server zu löschen. Dies ist nützlich, um veraltete Sitzungen aus dem Cache des Brokers zu entfernen.

Wenn beispielsweise ein NICE DCV Server gestoppt wurde, die Sitzungen aber immer noch auf dem Broker registriert sind, verwenden Sie dieses Flag, um die Sitzungen aus dem Cache des Brokers zu löschen.

Denken Sie daran, dass die Sitzung, wenn sie noch aktiv ist, vom Broker erneut zwischengespeichert wird.

Zulässige Werte: `true` | `false`

Typ: Boolesch

Erforderlich: Nein

Antwortparameter

`SessionId`

Die ID der Sitzung

`State`

Wird nur zurückgegeben, wenn die Sitzungen erfolgreich gelöscht wurden. Zeigt den aktuellen Status der Sitzung an. Wenn die Anforderung erfolgreich abgeschlossen wurde, wechselt die Sitzung in den `DELETING` Status. Es kann einige Minuten dauern, bis die Sitzung gelöscht ist. Nach dem Löschen wechselt der Status von `DELETING` zu`DELETED`.

`FailureReason`

Wird nur zurückgegeben, wenn einige Sitzungen nicht gelöscht werden konnten. Zeigt an, warum die Sitzung nicht gelöscht werden konnte.

Beispiel

Python

Anforderung

Im folgenden Beispiel werden zwei Sitzungen gelöscht — eine Sitzung mit der ID`SessionId123`, die gehört`user1`, und eine Sitzung mit der ID, `SessionIdabc` die Eigentümer ist. `user99`

```
from swagger_client.models.delete_session_request_data import
 DeleteSessionRequestData
```

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

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

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

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

Antwort

Im Folgenden finden Sie ein Beispiel für die Ausgabe. `SessionId123`wurde erfolgreich gelöscht, `SessionIdabc` konnte aber nicht gelöscht werden.

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

GetSessionConnectionData

Ruft Verbindungsinformationen für die Verbindung eines bestimmten Benutzers zu einer bestimmten NICE DCV Sitzung ab.

Themen

- Anforderungsparameter
- Antwortparameter
- Zusätzliche Informationen
- Beispiel

Anforderungsparameter

`SessionId`

Die ID der Sitzung, für die Verbindungsinformationen angezeigt werden sollen.

Typ: Zeichenfolge

Erforderlich: Ja

`User`

Der Name des Benutzers, für den Verbindungsinformationen angezeigt werden sollen.

Typ: Zeichenfolge

Erforderlich: Ja

Antwortparameter

`Id`

Die eindeutige ID der Sitzung.

`Name`

Der Name der Sitzung.

`Owner`

Der Besitzer der Sitzung.

`Server`

Informationen über den Server, auf dem die Sitzung läuft. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`Ip`

Die IP-Adresse des NICE DCV Serverhosts.

`Hostname`

Der Hostname des NICE DCV Serverhosts.

`Port`

Der Port, über den der NICE DCV Server mit NICE DCV den Clients kommuniziert.

`Endpoints`

Informationen zu den NICE DCV Serverendpunkten. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`IpAddress`

Die IP-Adresse des Serverendpunkts.

`Port`

Der Port des Serverendpunkts.

`Protocol`

Das vom Serverendpunkt verwendete Protokoll. Mögliche Werte sind:

- `HTTP`— Der Endpunkt verwendet das Protokoll WebSocket (TCP).
- `QUIC`— Der Endpunkt verwendet das QUIC (UDP) -Protokoll.

`WebUrlPath`

Der URL Webpfad des Serverendpunkts. Nur für das HTTP Protokoll verfügbar.

`WebUrlPath`

Der Pfad zur Konfigurationsdatei des NICE DCV Servers.

Tags

Die dem Server zugewiesenen Tags. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

Key

Der Tag-Schlüssel.

Value

Der Tag-Wert.

Type

Die Art der Sitzung.

State

Der aktuelle Status der Sitzung. Die möglichen Werte sind:

- `CREATING`- Der Broker ist dabei, die Sitzung zu erstellen.
- `READY`- Die Sitzung ist bereit, Client-Verbindungen anzunehmen.
- `DELETING`- Die Sitzung wird gelöscht.
- `DELETED`- Die Sitzung wurde gelöscht.
- `UNKNOWN`- Der Status der Sitzung konnte nicht ermittelt werden. Der Broker und der Agent können möglicherweise nicht kommunizieren.

CreationTime

Datum und Uhrzeit der Erstellung der Sitzung.

LastDisconnectionTime

Datum und Uhrzeit der letzten Verbindungsunterbrechung des Clients.

NumOfConnections

Die Anzahl der gleichzeitigen Verbindungen, die der Benutzer zur Sitzung hat.

`ConnectionToken`

Das Authentifizierungstoken, das für die Verbindung mit der Sitzung verwendet wird.

Zusätzliche Informationen

Die daraus gewonnenen Informationen API können an einen NICE DCV Client weitergegeben werden, um eine Verbindung zur NICE DCV Sitzung herzustellen.

Im Fall des NICE DCV Webclients können Sie einen erstellenURL, der im Browser geöffnet werden kann. Der URL hat das folgende Format:

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

Im Fall des NICE DCV nativen Clients können Sie einen URL mit dem `dcv://` Schema erstellen. Wenn der NICE DCV native Client installiert ist, registriert er sich selbst beim System als Handler für `dcv://`URLs. Der URL hat das folgende Format:

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

> **Note**
>
> Wenn Sie Amazon verwendenEC2, sollte die IP-Adresse die öffentliche sein. Wenn in Ihrer Konfiguration NICE DCV Hosts hinter einem Gateway stehen, geben Sie die Gateway-Adresse an und nicht die Adresse, die von zurückgegeben wurde SessionConnectionData API.

Beispiel

Python

Anforderung

Im folgenden Beispiel werden Verbindungsinformationen für einen Benutzer mit dem Benutzernamen von `user1` und einer Sitzung mit der ID von abgerufen`sessionId12345`.

```
def get_session_connection_api():
```

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


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


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


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

Antwort

Im Folgenden finden Sie ein Beispiel für die Ausgabe.

```
{
    "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

Ruft Screenshots von einer oder mehreren NICE DCV Sitzungen ab.

Der Bilddateityp und die Auflösung des Screenshots hängen von der Session Manager Broker-Konfiguration ab. Um den Bilddateityp zu ändern, konfigurieren Sie den `session-screenshot-format` Parameter. Um die Auflösung zu ändern, konfigurieren Sie die `session-screenshot-max-height` Parameter `session-screenshot-max-width` und. Weitere Informationen finden Sie in der Broker-Konfigurationsdatei im NICEDCVSession Manager-Administratorhandbuch.

Themen

- Anforderungsparameter
- Antwortparameter
- Beispiel

Anforderungsparameter

`SessionId`

Die ID der NICE DCV Sitzung, von der der Screenshot abgerufen werden soll.

Typ: Zeichenfolge

Erforderlich: Ja

Antwortparameter

`RequestId`

Die eindeutige ID der Anfrage.

`SuccessfulList`

Informationen zu den erfolgreichen Screenshots. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

`SessionScreenshot`

Informationen zu den Screenshots. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

SessionId

Die ID der NICE DCV Sitzung, aus der der Screenshot aufgenommen wurde.

Images

Informationen zu den Bildern. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

Format

Das Format des Bilds. Mögliche Werte sind u. a. `jpeg` und `png`.

Data

Das im Base64-kodierten Format des Screenshot-Bildes.

CreationTime

Datum und Uhrzeit der Aufnahme des Screenshots.

Primary

Gibt an, ob es sich bei dem Screenshot um das Hauptdisplay der NICE DCV Sitzung handelt.

UnsuccessfulList

Informationen zu den erfolglosen Screenshots. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

GetSesionScreenshotRequestData

Die ursprüngliche Anfrage, die fehlgeschlagen ist.

SessionId

Die ID der NICE DCV Sitzung, aus der der Screenshot aufgenommen werden sollte.

FailureReason

Der Grund für den Fehlschlag.

Beispiel

Python

Anforderung

Im folgenden Beispiel werden Screenshots aus zwei Sitzungen abgerufen (`sessionId1`und`sessionId2`). Die Sitzung `sessionId2` ist nicht vorhanden und führt zu einem Fehler.

```
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"])
```

Antwort

Im Folgenden finden Sie ein Beispiel für die Ausgabe.

```
{
    "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

Öffnet einen oder mehrere NICE DCV Server. Bevor Sie NICE DCV Sitzungen auf einem NICE DCV Server erstellen können, müssen Sie den Serverstatus auf Öffnen ändern. Nachdem der NICE DCV Server geöffnet ist, können Sie NICE DCV Sitzungen auf dem Server erstellen.

Themen

- Anforderungsparameter
- Antwortparameter
- Beispiel

Anforderungsparameter

`ServerId`

Die ID des Servers, der geöffnet werden soll.

Typ: Zeichenfolge

Erforderlich: Ja

Antwortparameter

`RequestId`

Die eindeutige ID der Anfrage.

SuccessfulList

Informationen über die NICE DCV Server, die erfolgreich geöffnet wurden. Diese Datenstruktur umfasst den folgenden verschachtelten Antwortparameter:

ServerId

Die ID des Servers, der erfolgreich geöffnet wurde.

UnsuccessfulList

Informationen zu den NICE DCV Servern, die nicht geöffnet werden konnten. Diese Datenstruktur umfasst die folgenden verschachtelten Antwortparameter:

OpenServerRequestData

Informationen über die ursprüngliche Anfrage, die fehlgeschlagen ist. Diese Datenstruktur umfasst den folgenden verschachtelten Antwortparameter:

ServerId

Die ID des NICE DCV Servers, der nicht geöffnet werden konnte.

FailureCode

Der Code des Fehlers.

FailureReason

Der Grund für den Fehlschlag.

Beispiel

Python

Anforderung

Im folgenden Beispiel NICE DCV werden zwei Server (`serverId1`und`serverId2`) geöffnet.

```
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"])
```

Antwort

Das Folgende ist die Beispielausgabe.

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

UpdateSessionPermissions

Aktualisiert die Benutzerberechtigungen für eine bestimmte NICE DCV Sitzung.

Themen

- Anforderungsparameter

- Antwortparameter
- Beispiel

Anforderungsparameter

`SessionId`

Die ID der Sitzung, für die die Berechtigungen geändert werden sollen.

Typ: Zeichenfolge

Erforderlich: Ja

`Owner`

Der Besitzer der Sitzung, für die die Berechtigungen geändert werden sollen.

Typ: Zeichenfolge

Erforderlich: Ja

`PermissionFile`

Der Base64-kodierte Inhalt der zu verwendenden Berechtigungsdatei. Weitere Informationen finden Sie unter Konfiguration der NICE DCV Autorisierung im Administratorhandbuch. NICE DCV

Typ: Zeichenfolge

Erforderlich: Ja

Antwortparameter

`SessionId`

Die ID der Sitzung.

Beispiel

Python

Anforderung

Im folgenden Beispiel werden neue Berechtigungen für eine Sitzung mit der Sitzungs-ID von festgelegt`SessionId1897`.

```
from swagger_client.models.update_session_permissions_request_data import
 UpdateSessionPermissionsRequestData

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

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

Antwort

Das Folgende ist die Beispielausgabe.

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

Versionshinweise und Dokumentenverlauf für NICE DCV Session Manager

Diese Seite enthält die Versionshinweise und den Dokumentverlauf für NICE DCV Session Manager.

Themen

- NICEDCVVersionshinweise zu Session Manager
- Dokumentverlauf

NICEDCVVersionshinweise zu Session Manager

Dieser Abschnitt bietet einen Überblick über die wichtigsten Updates, Feature-Releases und Bugfixes für NICE DCV Session Manager. Alle Updates sind nach Veröffentlichungsdatum geordnet. Wir aktualisieren die Dokumentation regelmäßig, um das Feedback zu berücksichtigen, das Sie uns senden.

Themen

- 2023.1-17652 — 1. August 2024
- 2023.1-16388 — 26. Juni 2024
- 2023.1 — 9. November 2023
- 2023.0-15065 — 4. Mai 2023
- 2023.0-14852 — 28. März 2023
- 2022.2-13907 — 11. November 2022
- 2022.1-13067 — 29. Juni 2022
- 2022.0-11952 — 23. Februar 2022
- 2021.3-11591 — 20. Dezember 2021
- 2021.2-11445 — 18. November 2021
- 2021.2-11190 — 11. Oktober 2021
- 2021.2-11042 — 01. September 2021
- 2021.1-10557 — 31. Mai 2021
- 2021.0-10242 — 12. April 2021
- 2020.2-9662 — 04. Dezember 2020

- 2020.2-9508 — 11. November 2020

2023.1-17652 — 1. August 2024

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 426 • Makler: 748 • CLI: 140	• Fehlerbehebungen und Leistungsverbesserungen.

2023.1-16388 — 26. Juni 2024

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 417 • Makler: 748 • CLI: 140	• Es wurde ein Fehler behoben, durch den Speicher fälschlicherweise als TB und nicht als GB angezeigt wurde. • Fehlerbehebungen und Leistungsverbesserungen.

2023.1 — 9. November 2023

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 410 • Makler: 732 • CLI: 140	• Fehlerbehebungen und Leistungsverbesserungen

2023.0-15065 — 4. Mai 2023

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 392	• Unterstützung für Red Hat Enterprise Linux 9, Rocky Linux 9 und CentOS Stream 9 auf ARM Plattformen hinzugefügt.

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 675 • CLI: 132	

2023.0-14852 — 28. März 2023

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 392 • Makler: 642 • CLI: 132	• Unterstützung für Red Hat Enterprise Linux 9, Rocky Linux 9 und CentOS Stream 9 hinzugefügt.

2022.2-13907 — 11. November 2022

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 382 • Makler: 612 • CLI: 123	• `DescribeSessions` Als Antwort wurde ein `Substate` Feld hinzugefügt. • Es wurde ein Problem behoben, das CLI dazu führen konnte, dass der je nach verwendetem Broker keine Verbindung zum Broker herstellen konnte. URL

2022.1-13067 — 29. Juni 2022

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 355 • Makler: 592 • CLI: 114	• Unterstützung für die Ausführung des Brokers AWS auf Graviton-Instances hinzugefügt. • Agenten- und Broker-Unterstützung für Ubuntu 22.04 hinzugefügt.

2022.0-11952 — 23. Februar 2022

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 341 • Makler: 520 • CLI: 112	• Dem Agenten wurde die Funktion zur Rotation von Protokollen hinzugefügt. • Konfigurationsparameter hinzugefügt, um Java Home im Broker festzulegen. • Die Übertragung von Daten vom Cache auf die Festplatte im Broker wurde verbessert. • Die URL Validierung in der wurde behoben. CLI

2021.3-11591 — 20. Dezember 2021

Build-Nummern	Neue Features
• Makler: 307 • Makler: 453 • CLI: 92	• Unterstützung für die Integration mit dem NICE DCV Connection Gateway hinzugefügt. • Broker-Unterstützung für Ubuntu 18.04 und Ubuntu 20.04 hinzugefügt.

2021.2-11445 — 18. November 2021

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 288 • Makler: 413 • CLI: 54	• Ein Problem mit der Überprüfung von Anmeldenamen, die eine Windows-Domäne enthalten, wurde behoben.

2021.2-11190 — 11. Oktober 2021

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 254 • Makler: 413 • CLI: 54	• Es wurde ein Problem in der Befehlszeilenschnittstelle behoben, das das Starten von Windows-Sitzungen verhinderte.

2021.2-11042 — 01. September 2021

Build-Nummern	Neue Features	Änderungen und Fehlerbehebungen
• Makler: 254 • Makler: 413 • CLI: 37	• NICEDCVSession Manager bietet jetzt Unterstützung für die Befehlszeilenschnittstelle (CLI). Sie können NICE DCV Sitzungen im erstellen und verwaltenCLI, anstatt sie aufzurufenAPIs. • NICEDCVMit Session Manager wurde Broker-Datenpersistenz eingeführt. Für eine höhere Verfügbarkeit können Broker Serverstatusinformationen in einem externen Datenspeicher speichern und die Daten beim Start wiederherstellen.	• Bei der Registrierung eines externen Autorisierungsservers können Sie jetzt den Algorithmus angeben, den der Autorisierungsserver zum Signieren JSON formatierter Web-Token verwendet. Mit dieser Änderung können Sie Azure AD als externen Autorisierungsserver verwenden.

2021.1-10557 — 31. Mai 2021

Build-Nummern	Neue Features	Änderungen und Fehlerbehebungen
• Makler: 214 • Makler: 365	• NICEDCVSession Manager hat Unterstützung für Eingabeparameter hinzugefügt, die an die Autorun-Datei unter Linux übergeben werden. • Servereigenschaften können jetzt als Anforderungen an die CreateSessionsAPIübergeben werden.	• Wir haben ein Problem mit der Autorun-Datei unter Windows behoben.

2021.0-10242 — 12. April 2021

Build-Nummern	Änderungen und Fehlerbehebungen

• Makler: 183
• Makler: 318
	• NICEDCVSession Manager hat die folgenden Neuerungen eingeführ tAPIs:
	  • OpenServers
	  • CloseServers
	  • DescribeServers
	  • GetSessionScreenshots
	• Außerdem wurden die folgenden neuen Konfigurationsparameter eingeführt:
	  • Broker-Parameter: `session-screenshot-max-widt h session-screenshot-max-height ,session-s creenshot-format ,create-sessions-queue-max-s ize , undcreate-sessions-queue-max-time-seconds .`
	  • Agentenparameter: `agent.autorun_folder max_virtu al_sessions , undmax_concurrent_sessions_per _user .`

Build-Nummern	Änderungen und Fehlerbehebungen
	Agentenparameter: `agent.autorun_folder` `max_virtual_sessions` , und`max_concurrent_sessions_per_user` . Agentenparameter: `agent.autorun_folder` `max_virtual_sessions` , und`max_concurrent_sessions_per_user` .

2020.2-9662 — 04. Dezember 2020

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 114 • Makler: 211	• Wir haben ein Problem mit den automatisch generierten TLS Zertifikaten behoben, das den Start des Brokers verhinderte.

2020.2-9508 — 11. November 2020

Build-Nummern	Änderungen und Fehlerbehebungen
• Makler: 78 • Makler: 183	• Die erste Version von NICE DCV Session Manager.

Dokumentverlauf

In der folgenden Tabelle wird die Dokumentation für diese Version von NICE DCV Session Manager beschrieben.

Änderung	Beschreibung	Datum
NICEDCVVersion	NICEDCVSession Manager wurde für 2023.1-17652 aktualisiert. NICE DCV	1. August 2024

Änderung	Beschreibung	Datum
2023.1-17652	Weitere Informationen finden Sie unter 2023.1-17652 — 1. August 2024.	
NICEDCVAusführung 2023.1-16388	NICEDCVSession Manager wurde für 2023.1-16388 aktualisiert. NICE DCV Weitere Informationen finden Sie unter 2023.1-16388 — 26. Juni 2024.	26. Juni 2024
NICEDCVAusführung 2023.1	NICEDCVSession Manager wurde für NICE DCV 2023.1 aktualisiert. Weitere Informationen finden Sie unter 2023.1 — 9. November 2023.	9. November 2023
NICEDCVVersion 2023.0	NICEDCVSession Manager wurde für NICE DCV 2023.0 aktualisiert. Weitere Informationen finden Sie unter 2023.0-14852 — 28. März 2023.	28. März 2023
NICEDCVVersion 2022.2	NICEDCVSession Manager wurde für NICE DCV 2022.2 aktualisiert. Weitere Informationen finden Sie unter 2022.2-13907 — 11. November 2022.	11. November 2022
NICEDCVVersion 2022.1	NICEDCVSession Manager wurde für NICE DCV 2022.1 aktualisiert. Weitere Informationen finden Sie unter 2022.1-13067 — 29. Juni 2022.	29. Juni 2022
NICEDCVVersion 2022.0	NICEDCVSession Manager wurde für NICE DCV 2022.0 aktualisiert. Weitere Informationen finden Sie unter 2022.0-11952 — 23. Februar 2022.	23. Februar 2022
NICEDCVVersion 2021.3	NICEDCVSession Manager wurde für NICE DCV 2021.3 aktualisiert. Weitere Informationen finden Sie unter 2021.3-11591 — 20. Dezember 2021.	20. Dezember 2021

Änderung	Beschreibung	Datum
NICEDCVVersion 2021.2	NICEDCVSession Manager wurde für NICE DCV 2021.2 aktualisiert. Weitere Informationen finden Sie unter 2021.2-11042 — 01. September 2021.	01. September 2021
NICEDCVAusführung 2021.1	NICEDCVSession Manager wurde für NICE DCV 2021.1 aktualisiert. Weitere Informationen finden Sie unter 2021.1-10557 — 31. Mai 2021.	31. Mai 2021
NICEDCVVersion 2021.0	NICEDCVSession Manager wurde für NICE DCV 2021.0 aktualisiert. Weitere Informationen finden Sie unter 2021.0-10242 — 12. April 2021.	12. April 2021
Erste Version von NICE DCV Session Manager	Die erste Veröffentlichung dieses Inhalts.	11. November 2020

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.