aws

管理者ガイド

NICE DCV セッションマネージャー

NICE DCV セッションマネージャー: 管理者ガイド

Table of Contents

セッションマネージャーとは 1
セッションマネージャーの仕組み 1
機能 3
制限事項 3
料金 4
要件 4
ネットワーク要件および接続要件 5
セットアップ 7
ステップ 1: NICE DCV サーバーを準備する 7
ステップ 2: ブローカーをセットアップする 8
ステップ 3: エージェントをセットアップする 10
ステップ 4: NICE DCV サーバーを設定する 15
ステップ 5: インストールを検証する 17
エージェントの検証 17
ブローカーの検証 18
設定 19
セッションマネージャーのスケーリング 19
ステップ 1: インスタンスプロファイルを作成する 20
ステップ 2: ロードバランサーの SSL 証明書を準備する 21
ステップ 3: ブローカーの Application Load Balancer を作成する 21
ステップ 4: ブローカーを起動する 23
ステップ 5: エージェントの Application Load Balancer を作成する 24
ステップ 6: エージェントの起動 25
タグの使用 27
外部認可サーバーの設定 28
ブローカー永続性の設定 33
DynamoDB で永続化されるようにブローカーを設定する 34
MariaDB/MySQL で永続化させるためにブローカーを設定する 35
NICE DCV 接続ゲートウェイとの統合 36
セッションマネージャーブローカーを NICE DCV 接続ゲートウェイのセッションリゾルバーとして設定 37
オプション - TLS クライアント認証の有効化 38
NICE DCV サーバー - DNS マッピング 39
Amazon CloudWatch との統合 41

アップグレード 43
NICE DCV セッションマネージャーエージェントのアップグレード 43
NICE DCV セッションマネージャーブローカーのアップグレード 45
ブローカー CLI リファレンス 48
register-auth-server 49
構文 49
オプション 49
例 49
list-auth-servers 50
構文 49
出力 50
例 49
unregister-auth-server 51
構文 49
オプション 49
出力 50
例 49
register-api-client 52
構文 49
オプション 49
出力 50
例 49
describe-api-clients 53
構文 49
出力 50
例 49
unregister-api-client 55
構文 49
オプション 49
例 49
renew-auth-server-api-key 56
構文 49
例 49
generate-software-statement 56
構文 49
出力 50

例 ... 49
describe-software-statements ... 58
構文 ... 49
出力 ... 50
例 ... 49
deactivate-software-statement ... 59
構文 ... 49
オプション ... 49
例 ... 49
describe-agent-clients ... 60
構文 ... 49
出力 ... 50
例 ... 49
unregister-agent-client ... 61
構文 ... 49
オプション ... 49
例 ... 49
register-server-dns-mappings ... 62
構文 ... 49
オプション ... 49
例 ... 49
describe-server-dns-mappings ... 63
構文 ... 49
出力 ... 50
例 ... 49
設定ファイルリファレンス ... 65
ブローカー設定ファイル ... 65
エージェント設定ファイル ... 81
リリースノートとドキュメント履歴 ... 87
リリースノート ... 87
2023.1-16388 — 2024 年 6 月 26 日 ... 88
2023.1— 2023 年 11 月 9 日 ... 88
2023.0-15065— 2023 年 5 月 4 日 ... 88
2023.0-14852— 2023 年 3 月 28 日 ... 88
2022.2-13907— 2022 年 11 月 11 日 ... 89
2022.1-13067— 2022 年 6 月 29 日 ... 89

2022.0-11952 — 2022 年 2 月 23 日 89
2021.3-11591 — 2021 年 12 月 20 日 90
2021.2-11445 — 2021 年 11 月 18 日 90
2021.2-11190 — 2021 年 10 月 11 日 90
2021.2-11042 — 2021 年 9 月 1 日 90
2021.1-10557 — 2021 年 5 月 31 日 91
2021.0-10242 — 2021 年 4 月 12 日 91
2020.2-9662 — 2020 年 12 月 4 日 92
..... 92
ドキュメント履歴 93
..... xcv

NICE DCV セッションマネージャーとは

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

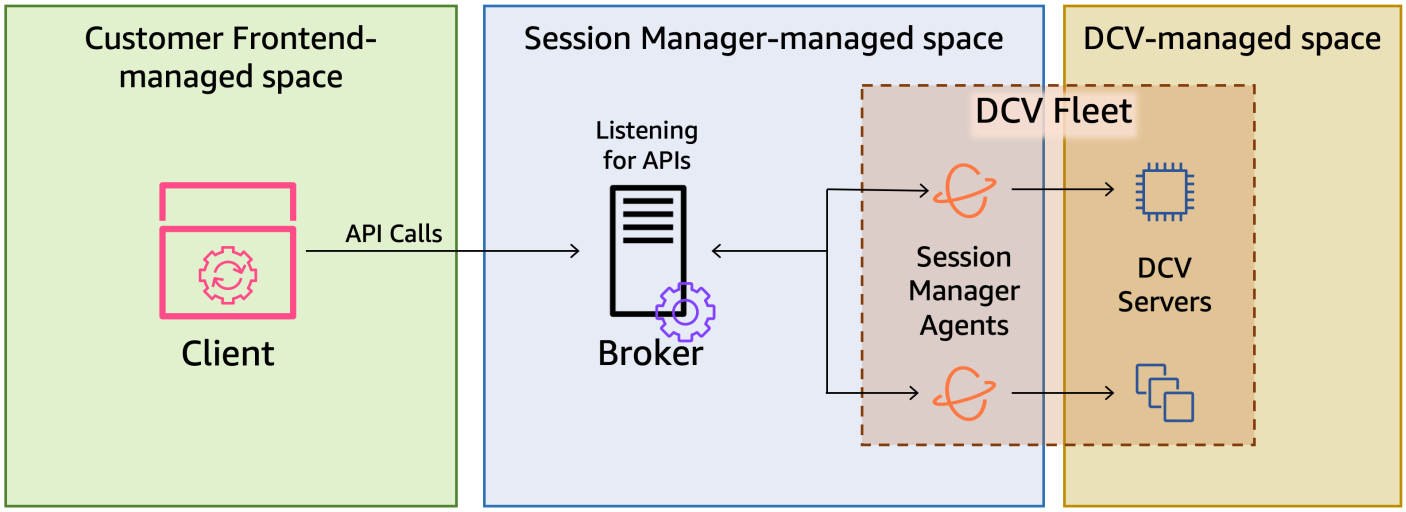
このガイドでは、セッションマネージャーのエージェントとブローカーをインストールして設定する方法について説明します。セッションマネージャー API の詳しい使用方法については、「NICE DCV セッションマネージャーデベロッパーガイド」を参照してください。

トピック

- セッションマネージャーの仕組み
- 機能
- 制限事項
- 料金
- NICE DCV セッションマネージャーの要件

セッションマネージャーの仕組み

次の図は、セッションマネージャーの高レベルコンポーネントを示しています。



ブローカー

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

エージェント

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

API

セッションマネージャーでは、一連の NICE DCV サーバーで NICE DCV セッションを管理するために使用できる REST アプリケーションプログラムインターフェイス (API) のセットが公開されます。API はブローカーでホストされて公開されます。デベロッパーは API を呼び出せるカスタムのセッション管理クライアントを構築できます。

クライアント

クライアントは、ブローカーにより公開されるセッションマネージャー API を呼び出すために開発されるフロントエンドのアプリケーションまたはポータルです。エンドユーザーは、クライアントを使用して、フリート内の NICE DCV サーバーでホストされるセッションを管理します。

アクセストークン

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

クライアント API

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

NICE DCV セッション

NICE DCV セッションは、クライアントによる接続が可能な NICE DCV サーバーで作成する必要があります。クライアントは、アクティブなセッションがある場合にのみ NICE DCV サーバー

に接続できます。NICE DCV ではコンソールセッションと仮想セッションがサポートされています。セッションマネージャー API を使用して、NICE DCV セッションのライフサイクルを管理します。NICE DCV セッションは、次のいずれかの状態になります。

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

機能

DCV セッションマネージャーには以下の特徴があります。

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

制限事項

セッションマネージャーには、リソースのプロビジョニング機能はありません。NICE DCV を Amazon EC2 インスタンスで実行している場合は、インフラストラクチャのスケーリングを管理するために Amazon EC2 Auto Scaling などの追加の AWS サービスの使用が必要になる可能性があります。

料金

EC2 インスタンスを実行する AWS のお客様の場合は、セッションマネージャーを無料でご利用いただけます。

オンプレミスのお客様の場合は、NICE DCV Plus または DCV Professional Plus のライセンスを取得していただく必要があります。NICE DCV Plus や NICE DCV Professional Plus のライセンスの購入方法については、NICE のウェブサイトの How to Buy (購入方法) を参照してください。お住まいの地域の NICE 販売代理店または再販業者が掲載されています。すべてのオンプレミスのお客様に DCV セッションマネージャーをお試しいただけるように、ライセンス要件は NICE DCV バージョン 2021.0 以降のみに適用されます。

詳細については、「NICE DCV 管理者ガイド」の「NICE DCV サーバーのライセンス」を参照してください。

NICE DCV セッションマネージャーの要件

NICE DCV セッションマネージャーのエージェントとブローカーには、次の要件があります。

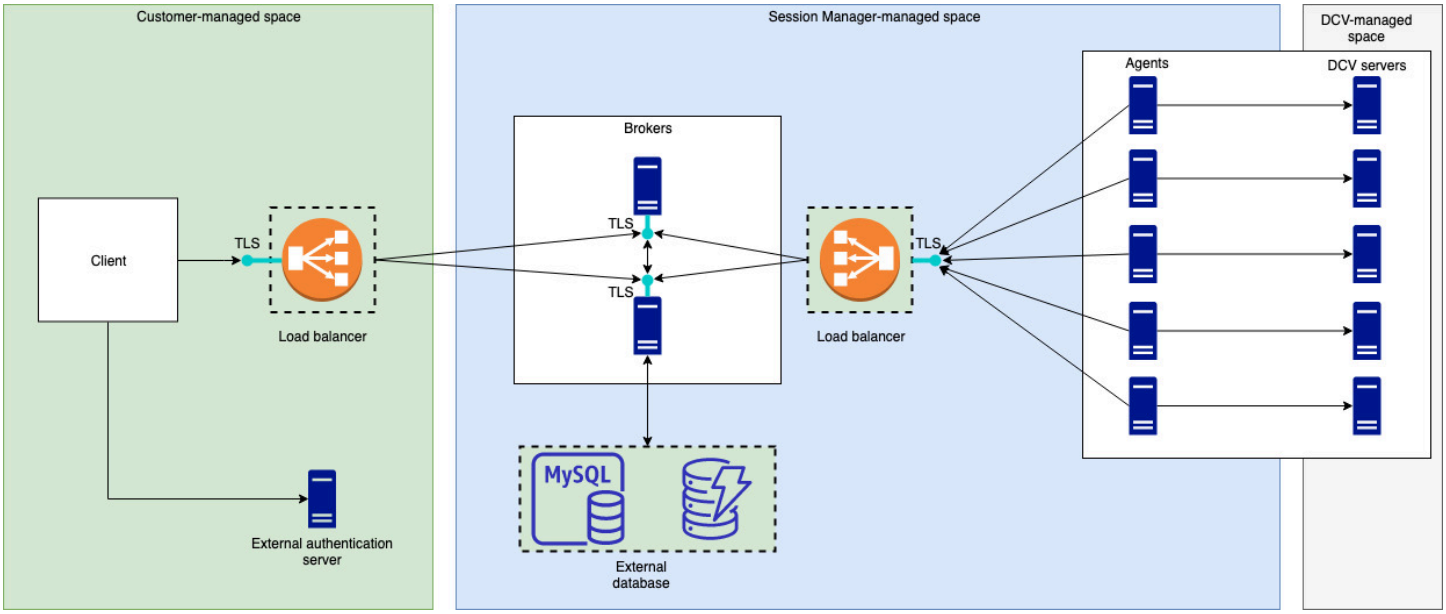
	ブローカー	エージェント

オペレーティングシステム	• Amazon Linux 2
	• CentOS 7.6 以降
	• CentOS Stream 8
	• RHEL 7.6 以降
	• RHEL 8.x
	• Rocky Linux 8.5 以降
	• Ubuntu 20.04
	• Ubuntu 22.04
		• Windows
		  • Windows Server 2019
		  • Windows Server 2016
		  • Windows Server 2012 R2
		• Linux サーバー
		  • Amazon Linux 2
		  • CentOS 7.6 以降
		  • CentOS Stream 8
		  • RHEL 7.6 以降
		  • RHEL 8.x
		  • Rocky Linux 8.5 以降
		  • Ubuntu 20.04
		  • Ubuntu 22.04

	ブローカー	エージェント
		• SP4 以降の SUSE Linux Enterprise 12 • SUSE Linux Enterprise 15
アーキテクチャ	• 64 ビット x86 • 64 ビット ARM	• 64 ビット x86 • 64 ビット ARM (Amazon Linux 2、CentOS 7.x/8.x、RHEL 7.x/8.x のみ) • 64 ビット ARM (Ubuntu 22.04)
メモリ	8 GB	4 GB
NICE DCV バージョン	NICE DCV 2020.2 以降	NICE DCV 2020.2 以降
その他の要件	Java 11	-

ネットワーク要件および接続要件

次の図は、セッションマネージャーのネットワーク要件と接続要件の高レベルの概要を示しています。



ブローカーは別のホストにインストールする必要がありますが、NICE DCV サーバーのエージェントとのネットワーク接続が必要になります。可用性を向上させるために複数のブローカーを使用する場合は、各ブローカーを別々のホストにインストールして設定し、1 つ以上のロードバランサーを使用して、クライアントとブローカー間とブローカーとエージェント間のトラフィックを管理する必要があります。ブローカーは、NICE DCV サーバーとセッションに関する情報を交換するために相互通信が可能な状態でなければなりません。ブローカーでは、キーとステータスデータを外部データベースに保存し、再起動後または終了後にこの情報を入手可能な状態にすることができます。重要なブローカー情報を外部データベースに保存することで、重要なブローカー情報を失うリスクを軽減できます。この情報を後で復元することはできません。情報の保持を選択した場合は、外部データベースをセットアップしてブローカーを設定する必要があります。DynamoDB、MariaDB、MySQL がサポートされます。設定パラメータはブローカー設定ファイルで確認できます。

エージェントは、ブローカーとの安全かつ永続的な双方向 HTTPS 接続を開始できる状態でなければなりません。

クライアント、またはフロントエンドアプリケーションは、API を呼び出すためにブローカーにアクセスできる状態でなければなりません。クライアントは認証サーバーにアクセスすることもできます。

NICE DCV セッションマネージャーのセットアップ

次のセクションでは、単一のブローカーと複数のエージェントを使用してセッションマネージャーをインストールする方法について説明します。複数のブローカーを使用して、スケーラビリティとパフォーマンスを向上させることができます。詳細については、「セッションマネージャーのスケーリング」を参照してください。

NICE DCV セッションマネージャーをセットアップするには、次の手順を実行します。

ステップ

- ステップ 1: NICE DCV サーバーを準備する
- ステップ 2: NICE DCV セッションマネージャーブローカーをセットアップする
- ステップ 3: NICE DCV セッションマネージャーエージェントをセットアップする
- ステップ 4: ブローカーが認証サーバーとして使用されるように NICE DCV サーバーを設定する
- ステップ 5: インストールを検証する

ステップ 1: NICE DCV サーバーを準備する

セッションマネージャーを使用したい NICE DCV サーバーのフリートが必要です。NICE DCV サーバーのインストールの詳細については、「NICE DCV 管理者ガイド」の「NICE DCV サーバーのインストール」を参照してください。

Linux NICE DCV サーバーのセッションマネージャーでは dcvsmagent という名前のローカルサービスユーザーが使用されます。このユーザーは、セッションマネージャーエージェントのインストール時に自動的に作成されます。NICE DCV が他のユーザーに代わってアクションを実行できるように、このサービスユーザーに管理者権限を付与する必要があります。セッションマネージャーのサービスユーザー管理者権限を付与するには、次の手順を実行します。

Linux NICE DCV サーバーのローカルサービスユーザーを追加する方法

1. お好みのテキストエディタを使用して/etc/dcv/dcv.confを開きます。
2. administrators パラメータを [security] セクションに追加し、セッションマネージャーユーザーを指定します。例:

```
[security]
```

```
administrators=["dcvsmagent"]
```

3. ファイルを保存して閉じます。
4. NICE DCV サーバーを停止して再起動します。

セッションマネージャーは、NICE DCV サーバーにすでに存在するユーザーに代わって NICE DCV セッションの作成のみを実行できます。存在しないユーザーのセッションを作成するためにリクエストを出しても失敗します。したがって、対象の各エンドユーザーに NICE DCV サーバーの有効なシステムユーザーが含まれていることを確認する必要があります。

> **Tip**
>
> エージェントで複数のブローカーホストまたは NICE DCV サーバーを使用する場合は、完成された設定でホストの Amazon マシンイメージ (AMI) を作成し、AMI を使用して残りのブローカーサーバーと NICE DCV サーバーを起動することで、ブローカー 1 つと、1 つのエージェントを含む NICE DCV サーバーを 1 つだけ設定することをお勧めします。または、AWS Systems Manager を使用して、複数のインスタンスでリモートでコマンドを実行することもできます。

ステップ 2: NICE DCV セッションマネージャーブローカーをセットアップする

ブローカーを Linux ホストにインストールする必要があります。サポートされている Linux ディストリビューションの詳細については、「NICE DCV セッションマネージャーの要件」を参照してください。エージェントおよび NICE DCV サーバーホストとは別のホストにブローカーをインストールします。ホストは別のプライベートネットワークにインストールできますが、エージェントに接続して通信できる状態でなければなりません。

ブローカーをインストールして起動する方法

1. ブローカーをインストールするホストに接続します。
2. パッケージは安全な GPG 署名でデジタル署名されています。パッケージマネージャーでパッケージ署名を検証できるようにするには、NICE GPG キーをインポートする必要があります。NICE GPG キーをインポートするには、次のコマンドを実行します。

   - Amazon Linux 2、RHEL、CentOS、Rocky Linux

```
$ sudo rpm --import https://d1uj6qtbmh3dt5.cloudfront.net/NICE-GPG-KEY
```

- Ubuntu

```
$ wget https://d1uj6qtbmh3dt5.cloudfront.net/NICE-GPG-KEY gpg --import NICE-GPG-
KEY
```

3. インストールパッケージをダウンロードします。

- Amazon Linux 2、RHEL 7.x、CentOS 7.x

```
$ wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-
dcv-session-manager-broker-2023.1.410-1.el7.noarch.rpm
```

- RHEL 8.x、CentOS Stream 8、Rocky Linux 8.x

```
$ wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-
dcv-session-manager-broker-2023.1.410-1.el8.noarch.rpm
```

- Ubuntu 20.04

```
$ wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-
dcv-session-manager-broker_2023.1.410-1_all.ubuntu2004.deb
```

- Ubuntu 22.04

```
$ wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-
dcv-session-manager-broker_2023.1.410-1_all.ubuntu2204.deb
```

4. パッケージをインストールします。

- Amazon Linux 2、RHEL 7.x、CentOS 7.x

```
$ sudo yum install -y ./nice-dcv-session-manager-
broker-2023.1.410-1.el7.noarch.rpm
```

- RHEL 8.x、Stream CentOS 8、Rocky Linux 8.x

```
$ sudo yum install -y ./nice-dcv-session-manager-
broker-2023.1.410-1.el8.noarch.rpm
```

- Ubuntu 20.04

```
$ sudo apt install -y ./nice-dcv-session-manager-
broker_2023.1.410-1_all.ubuntu2004.deb
```

- Ubuntu 22.04

```
$ sudo apt install -y ./nice-dcv-session-manager-
broker_2023.1.410-1_all.ubuntu2204.deb
```

5. デフォルトの Java 環境バージョンが 11 であることを確認します。

```
$ java -version
```

バージョンが異なる場合、適切な Java バージョンを指定するように、ブローカーが使用する Java ホームディレクトリを明示的に設定できます。これは、ブローカー設定ファイルの broker-java-home パラメータを設定することで可能です。詳細については、「ブローカー設定ファイル」を参照してください。

6. ブローカーサービスを開始し、インスタンスを起動するたびにブローカーサービスが自動的に起動することを確認します。

```
$  sudo systemctl start dcv-session-manager-broker && sudo systemctl enable dcv-
session-manager-broker
```

7. ブローカーの自己署名証明書のコピーをユーザーディレクトリに入れます。これは次のステップでエージェントをインストールするときに必要になります。

```
sudo cp /var/lib/dcvsmbroker/security/dcvsmbroker_ca.pem $HOME
```

ステップ 3: NICE DCV セッションマネージャーエージェントをセットアップする

フリート内のすべての NICE DCV サーバーホストにエージェントをインストールする必要があります。Windows サーバーと Linux サーバーの両方にエージェントをインストールできます。サポートされるオペレーティングシステムの詳細については、「NICE DCV セッションマネージャーの要件」を参照してください。

前提条件

エージェントをインストールする前に、NICE DCV サーバーをホストにインストールする必要があります。

Linux host

> **Note**
>
> Session Manager Agent は、「要件」に記載されている Linux ディストリビューションとアーキテクチャで使用できます。
> 以下は、64 ビット x86 ホストにエージェントをインストールする手順です。*64 ## # ARM ######################x86_64 ############* aarch64*Ubuntu ### ##amd64 ############* arm64

Linux ホストにエージェントをインストールする方法

1. パッケージは安全な GPG 署名でデジタル署名されています。パッケージマネージャーでパッケージ署名を検証できるようにするには、NICE GPG キーをインポートする必要があります。NICE GPG キーをインポートするには、次のコマンドを実行します。

   - Amazon Linux 2、RHEL、CentOS、SUSE Linux Enterprise

     ```
     $ sudo rpm --import https://d1uj6qtbmh3dt5.cloudfront.net/NICE-GPG-KEY
     ```

   - Ubuntu

     ```
     $ wget https://d1uj6qtbmh3dt5.cloudfront.net/NICE-GPG-KEY
     ```

     ```
     $ gpg --import NICE-GPG-KEY
     ```

2. インストールパッケージをダウンロードします。

   - Amazon Linux 2、RHEL 7.x、CentOS 7.x

     ```
     $  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
     nice-dcv-session-manager-agent-2023.1.732-1.el7.x86_64.rpm
     ```

   - RHEL 8.x、CentOS Stream 8、Rocky Linux 8.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.el8.x86_64.rpm
```

- Ubuntu 20.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent_2023.1.732-1_amd64.ubuntu2004.deb
```

- Ubuntu 22.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent_2023.1.732-1_amd64.ubuntu2204.deb
```

- SUSE Linux Enterprise 12

```
$ curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$ curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.sles15.x86_64.rpm
```

3. パッケージをインストールします。

- Amazon Linux 2、RHEL 7.x、CentOS 7.x

```
$  sudo yum install -y ./nice-dcv-session-manager-
agent-2023.1.732-1.el7.x86_64.rpm
```

- RHEL 8.x、CentOS Stream 8、Rocky Linux 8.x

```
$  sudo yum install -y ./nice-dcv-session-manager-
agent-2023.1.732-1.el8.x86_64.rpm
```

- Ubuntu 20.04

```
$  sudo apt install ./nice-dcv-session-manager-
agent_2023.1.732-1_amd64.ubuntu2004.deb
```

- Ubuntu 22.04

```
$  sudo apt install ./nice-dcv-session-manager-
agent_2023.1.732-1_amd64.ubuntu2204.deb
```

- SUSE Linux Enterprise 12

```
$ sudo zypper install ./nice-dcv-session-manager-
agent-2023.1.732-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$ sudo zypper install ./nice-dcv-session-manager-
agent-2023.1.732-1.sles15.x86_64.rpm
```

4. ブローカーの自己署名証明書のコピー (前のステップでコピー済み) をエージェントの /etc/dcv-session-manager-agent/ ディレクトリに入れます。
5. 任意のテキストエディタを使用して /etc/dcv-session-manager-agent/agent.conf ファイルを開き、以下を実行します。

   - broker_host の場合、ブローカーがインストールされているホストの DNS 名を指定します。

     > **Important**
     >
     > ブローカーが Amazon EC2 インスタンスで実行されている場合は、broker_host に対してインスタンスのプライベート IPv4 アドレスを指定する必要があります。

   - (オプション) broker_port に対して、ブローカーとの通信用ポートを指定します。デフォルトでは、エージェントとブローカー間の通信はポート 8445 を介して行われます。別のポートを使用する必要がある場合のみ、これを選択します。変更する場合は、ブローカーで同じポートが使用されるように設定されていることを確認してください。
   - ca_file に対して、前のステップでコピーした証明書ファイルのフルパスを指定します。例:

     ```
     ca_file = '/etc/dcv-session-manager-agent/broker_cert.pem'
     ```

     TLS 検証を無効にする場合は、tls_strict を false に設定します。
6. ファイルを保存して閉じます。

7. エージェントを開始するには、次のコマンドを実行します。

```
$ sudo systemctl start dcv-session-manager-agent
```

Windows host

Windows ホストにエージェントをインストールする方法

1. エージェントインストーラーをダウンロードします。
2. インストーラーを実行します。ようこそ画面で、[Next] を選択します。
3. [EULA] 画面で、使用許諾書をしっかりと読んだ上で、条件に同意する場合は [I accept the terms] (同意します) を選択し、[Next] (次へ) を選択します。
4. インストールを開始するには [Install] (インストール) を選択します。
5. ブローカーの自己署名証明書のコピー (前のステップでコピー済み) をエージェントの `C:\Program Files\NICE\DCVSessionManagerAgent\conf\` フォルダに入れます。
6. 任意のテキストエディタを使用して `C:\Program Files\NICE\DCVSessionManagerAgent\conf\agent.conf` ファイルを開き、以下を実行します。

   - `broker_host` の場合、ブローカーがインストールされているホストの DNS 名を指定します。

     > **Important**
     >
     > ブローカーが Amazon EC2 インスタンスで実行されている場合は、`broker_host` に対してインスタンスのプライベート IPv4 アドレスを指定する必要があります。

   - (オプション) `broker_port` に対して、ブローカーとの通信用ポートを指定します。デフォルトでは、エージェントとブローカー間の通信はポート 8445 を介して行われます。別のポートを使用する必要がある場合のみ、これを選択します。変更する場合は、ブローカーで同じポートが使用されるように設定されていることを確認してください。
   - `ca_file` に対して、前のステップでコピーした証明書ファイルのフルパスを指定します。例:

     ```
     ca_file = 'C:\Program Files\NICE\DCVSessionManagerAgent\conf\broker_cert.pem'
     ```

TLS 検証を無効にする場合は、tls_strict を false に設定します。

7. ファイルを保存して閉じます。
8. エージェントサービスを停止して再起動し、変更を有効にします。コマンドプロンプトで、次のコマンドを入力します。

```
C:\> sc stop DcvSessionManagerAgentService
```

```
C:\> sc start DcvSessionManagerAgentService
```

ステップ 4: ブローカーが認証サーバーとして使用されるように NICE DCV サーバーを設定する

クライアント接続トークンの検証のためにブローカーが外部認証サーバーとして使用されるように NICE DCV サーバーを設定します。また、ブローカーの自己署名 CA を信頼するように NICE DCV サーバーを設定する必要もあります。

Linux NICE DCV server

Linux NICE DCV サーバーのローカルサービスユーザーを追加する方法

1. お好みのテキストエディタを使用して/etc/dcv/dcv.confを開きます。
2. ca-file パラメータと auth-token-verifier パラメータを [security] セクションに追加します。

   ca-file に対して、前のステップでホストにコピーしたブローカーの自己署名 CA へのパスを指定します。

   auth-token-verifier に対して、ブローカーのトークン検証の URL を https://*broker_ip_or_dns*:*port*/agent/validate-authentication-token 形式で指定します。ブローカーとエージェント間の通信に使用するポートを指定します。デフォルトでは 8445 です。Amazon EC2 インスタンスでブローカーを実行している場合は、プライベート DNS またはプライベート IP アドレスを使用する必要があります。

   例

```
[security]
```

```
ca-file="/etc/dcv-session-manager-agent/broker_cert.pem"
auth-token-verifier="https://my-sm-broker.com:8445/agent/validate-
authentication-token"
```

3. ファイルを保存して閉じます。
4. NICE DCV サーバーを停止して再起動します。詳細については、「NICE DCV 管理者ガイド」の「NICE DCV サーバーの停止」と「NICE DCV サーバーの起動」を参照してください。

Windows NICE DCV server

Windows NICE DCV サーバーの場合

1. Windows レジストリエディタを開き、HKEY_USERS/S-1-5-18/Software/GSettings/com/nicesoftware/dcv/security/ キーに移動します。
2. [ca-file] パラメータを開きます。値のデータに対して、前のステップでホストにコピーしたブローカーの自己署名 CA へのパスを指定します。

> **Note**
>
> パラメータが存在しない場合は、新しい文字列パラメータを作成して `ca-file` という名前を付けます。

3. パラメーターを開きます。auth-token-verifier値のデータ に対して、ブローカーのトークン検証の URL を `https://broker_ip_or_dns:port/agent/validate-authentication-token` 形式で指定します。ブローカーとエージェント間の通信に使用するポートを指定します。デフォルトでは 8445 です。Amazon EC2 インスタンスでブローカーを実行している場合は、プライベート DNS またはプライベート IP アドレスを使用する必要があります。

> **Note**
>
> パラメータが存在しない場合は、新しい文字列パラメータを作成して `auth-token-verifier` という名前を付けます。

4. [OK] を選択して Windows レジストリエディタを閉じます。

5. NICE DCV サーバーを停止して再起動します。詳細については、「NICE DCV 管理者ガイド」の「NICE DCV サーバーの停止」と「NICE DCV サーバーの起動」を参照してください。

ステップ 5: インストールを検証する

トピック

- エージェントの検証
- ブローカーの検証

エージェントの検証

ブローカーとエージェントをインストールしたら、エージェントが実行中であることと、ブローカーに接続できることを確認します。

Linux エージェントホスト

実行するコマンドはバージョンによって異なります。

- バージョン 2022.0 以降

  エージェントのホストから次のコマンドを実行します。」

```
$ grep 'sessionsUpdateResponse' /var/log/dcv-session-manager-agent/agent.log | tail
 -1 | grep -o success
```

- バージョン 2022.0 以前

  エージェントホストから次のコマンドを実行し、現在の年月日を指定します。

```
$ grep 'sessionsUpdateResponse' /var/log/dcv-session-manager-agent/
agent.log.yyyy-mm-dd | tail -1 | grep -o success
```

  例

```
$ grep 'sessionsUpdateResponse' /var/log/dcv-session-manager-agent/
agent.log.2020-11-19 | tail -1 | grep -o success
```

エージェントが実行中であり、ブローカーに接続できる場合、コマンドから success が返されます。

コマンドから別の出力が返された場合は、エージェントログファイルで詳細を調べます。ログファイルは /var/log/dcv-session-manager-agent/ にあります。

Windows エージェントホスト

C:\ProgramData\NICE\DCVSessionManagerAgent\log にあるエージェントログファイルを開きます。

ログファイルに次のような行が含まれている場合、エージェントは実行中であり、ブローカーに接続できます。

```
2020-11-02 12:38:03,996919  INFO ThreadId(05) dcvsessionmanageragent::agent:Processing
 broker message "{\n  \"sessionsUpdateResponse\" : {\n    \"requestId\" :
 \"69c24a3f5f6d4f6f83ffbb9f7dc6a3f4\",\n    \"result\" : {\n      \"success\" : true\n
   }\n  }\n}"
```

ログファイルにこのような行が含まれていない場合は、エラーがないかログファイルを調べます。

ブローカーの検証

ブローカーとエージェントをインストールしたら、ブローカーが実行中であることと、ユーザーとフロントエンドアプリケーションからアクセスできる状態であることを確認します。

ブローカーに到達できるコンピュータから、次のコマンドを実行します。

```
$ curl -X GET https://broker_host_ip:port/sessionConnectionData/aSession/aOwner --
insecure
```

検証が成功すると、ブローカーから以下が返されます。

```
{
    "error": "No authorization header"
}
```

NICE DCV セッションマネージャーの設定

このセクションでは、セッションマネージャーの高度な設定を行う方法について説明します。

トピック

- セッションマネージャーのスケーリング
- NICE DCV サーバーをターゲットにするためのタグの使用
- 外部認可サーバーの設定
- ブローカー永続性の設定
- NICE DCV 接続ゲートウェイとの統合
- Amazon CloudWatch との統合

セッションマネージャーのスケーリング

高可用性を実現してパフォーマンスを向上させるために、複数のエージェントとブローカーを使用するようにセッションマネージャーを設定することができます。複数のエージェントとブローカーを使用する場合は、エージェントとブローカーホストを 1 つだけインストールして設定し、これらのホストから Amazon マシンイメージ (AMI) を作成して、残りのホストを AMI から起動することをお勧めします。

セッションマネージャーではデフォルト設定で、追加の設定を行うことなく複数のエージェントの使用に対応しています。ただし、複数のブローカーを使用する場合は、ロードバランサーを使用して、フロントエンドクライアントとブローカー間、およびブローカーとエージェント間のトラフィックのバランスを取る必要があります。ロードバランサーのセットアップと設定は、ユーザーによって完全に所有され、かつ管理されます。

次のセクションでは、Application Load Balancer で複数のホストを使用するように Session Manager を設定する方法について説明します。

ステップ

- ステップ 1: インスタンスプロファイルを作成する
- ステップ 2: ロードバランサーの SSL 証明書を準備する
- ステップ 3: ブローカーの Application Load Balancer を作成する
- ステップ 4: ブローカーを起動する
- ステップ 5: エージェントの Application Load Balancer を作成する

- ステップ 6: エージェントの起動

ステップ 1: インスタンスプロファイルを作成する

Elastic Load Balancing API の使用許可を付与するブローカーホストとエージェントホストにインスタンスプロファイルをアタッチする必要があります。詳細については、Amazon EC2 ユーザーガイド の「Amazon EC2 の IAM ロール」を参照してください。

インスタンスプロファイルを作成するには

1. インスタンスプロファイルで使用するアクセス許可を定義する AWS Identity and Access Management (IAM) ロールを作成します。次の信頼ポリシーを使用します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

次に、以下のポリシーをアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
        "elasticloadbalancing:DescribeTargetHealth"
```

```
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

詳細については、「IAM ユーザーガイド」の「IAM ロールの作成」を参照してください。

2. 新しいインスタンスプロファイルを作成します。詳細については、「AWS CLI コマンドリファレンス」の「create-instance-profile」を参照してください。
3. インスタンスプロファイルに IAM ロールを追加します。詳細については、「AWS CLI コマンドリファレンス」の「add-role-to-instance-profile」を参照してください。
4. インスタンスプロファイルをブローカーホストアタッチします。詳細については、Amazon EC2 ユーザーガイド」の「IAM ロールをインスタンスにアタッチする」を参照してください。

ステップ 2: ロードバランサーの SSL 証明書を準備する

ロードバランサーに HTTPS を使用するときは、ロードバランサーに SSL 証明書をデプロイする必要があります。ターゲットにリクエストを送信する前に、ロードバランサーはこの証明書を使用して接続を終了し、クライアントからのリクエストを復号します。

SSL 証明書を準備するには

1. プライベート認証機関 (CA) AWS Certificate Manager プライベート認証機関 (ACM PCA) を作成します。詳細については、「 Certificate Manager Private Certificate Authority ユーザーガイド」の「CA の作成手順」を参照してください。 AWS
2. CA をインストールします。詳細については、「 Certificate Manager プライベート認証機関ユーザーガイド」の「ルート CA AWS 証明書のインストール」を参照してください。
3. CA によって署名された新しいプライベート証明書を要求します。ドメイン名については、*.*region*.elb.amazonaws.com を使用して、ロードバランサーを作成する予定のリージョンを指定します。詳細については、「 Certificate Manager Private Certificate Authority ユーザーガイド」の「プライベート証明書のリクエスト」を参照してください。 AWS

ステップ 3: ブローカーの Application Load Balancer を作成する

Application Load Balancer を作成して、フロントエンドクライアントとブローカー間のトラフィックのバランスを調整します。

ロードバランサーを作成する方法

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。

   ナビゲーションペインで、[Load Balancers] (ロードバランサー) を選択し、[Create Load Balancer] (ロードバランサーの作成) を選択します。[load balance type] (ロードバランサーのタイプ) で、[Application Load Balancer] を選択します。

2. [Step 1: Configure Load Balancer] で、以下の操作を行います。

   a. [Name] (名前) に、対象のロードバランサーを表現した説明的な名前を入力します。

   b. [Scheme] (スキーム) で、[internet-facing] (インターネット向け) を選択します。

   c. [Load Balancer Protocol] (ロードバランサーのプロトコル) で [HTTPS] を選択し、[Load Balancer Port] (ロードバランサーのポート) に 8443 を入力します。

   d. [VPC] で、使用する VPC を選択し、次にその VPC のサブネットを全て選択します。

   e. [次へ] をクリックします。

3. [Step 2: Configure Security Settings] (ステップ 2: セキュリティ設定を行う) で以下を実行します。

   a. [Certificate type] (証明書のタイプ) で、[Choose a certificate from ACM] (ACM から証明書を選択する) を選択します。

   b. [Certificate name] (証明書名) で、前の段階でリクエストしたプライベート証明書を選択します。

   c. [次へ] をクリックします。

4. [Step 3: Configure Security Groups] (ステップ 3: セキュリティグループを設定する) で、新しいセキュリティグループを作成するか、または、HTTPS とポート 8443 でフロントエンドクライアントとブローカー間のインバウンドトラフィックとアウトバウンドトラフィックが可能な既存のセキュリティグループを選択します。

   [次へ] をクリックします。

5. [Step 4: Configure Routing] (ステップ 4: ルーティングの設定) で、以下の操作を行います。

   a. [Target group] (ターゲットグループ) で、[New target group] (新しいターゲットグループ) を選択します。

   b. [名前] に、ターゲットグループの名前を入力します。

   c. [Target type] (ターゲットタイプ) で [Instance] (インスタンス) を選択します。

d. [Protocol] (プロトコル) で [HTTPS] を選択します。[Port (ポート)] に「8443」と入力します。[Protocol version] (プロトコルバージョン) で [HTTP1] を選択します。

e. ヘルスチェックの [Protocol] (プロトコル) で、[HTTPS] を選択し、[Path] (パス) に `/health` を入力します。

f. [次へ] をクリックします。

6. [Step 5: Register Targets] (ステップ 5: ターゲットを登録する) で [Next] (次へ) を選択します。

7. [Create] (作成) を選択します。

ステップ 4: ブローカーを起動する

初期ブローカーを作成し、ロードバランサーを使用するように設定して、ブローカーから AMI を作成し、AMI を使用して残りのブローカーを起動します。これにより、すべてのブローカーが、同一の CA と同一のロードバランサー設定を使用するように設定されます。

ブローカーの起動方法

1. 初期ブローカーホストを起動して設定します。ブローカーのインストールと設定の詳細については、「ステップ 2: NICE DCV セッションマネージャーブローカーをセットアップする」を参照してください。

> **Note**
>
> Application Load Balancer を使用しているため、ブローカーの自己署名証明書は必要ありません。

2. ブローカーに接続し、適切なテキストエディタを使用して `/etc/dcv-session-manager-broker/session-manager-broker.properties` ファイルを開き、以下を追加します。

a. 行の先頭にハッシュ (#) を付けることで `broker-to-broker-discovery-addresses` パラメータをコメントアウトします。

b. `broker-to-broker-discovery-aws-region` で、アプリケーションロードバランサーを作成したリージョンを入力します。

c. `broker-to-broker-discovery-aws-alb-target-group-arn` で、ブローカーのロードバランサーに関連付けられているターゲットグループの ARN を入力します。

d. ファイルを保存して閉じます。

3. ブローカーインスタンスを停止します。

4. 停止したブローカーインスタンスから AMI を作成します。詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「インスタンスからの Linux AMI の作成」を参照してください。
5. AMI を使用して残りのブローカーを起動します。
6. 作成したインスタンスプロファイルをすべてのブローカーインスタンスに割り当てます。
7. ブローカラーからブローカーへのネットワークトラフィックとブローカーからロードバランサーへのネットワークトラフィックを許可するセキュリティグループをすべてのブローカーインスタンスに割り当てます。ネットワークポートの詳細については、「ブローカー設定ファイル」をご参照ください。
8. すべてのブローカーインスタンスをブローカーロードバランサーのターゲットとして登録します。詳細については、「Application Load Balancer ユーザーガイド」の「ターゲットグループへのターゲットの登録」を参照してください。

ステップ 5: エージェントの Application Load Balancer を作成する

Application Load Balancer を作成してエージェントとブローカーのバランスを調整します。

ロードバランサーを作成する方法

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。

   ナビゲーションペインで、[Load Balancers] (ロードバランサー) を選択し、[Create Load Balancer] (ロードバランサーの作成) を選択します。[load balance type] (ロードバランサーのタイプ) で、[Application Load Balancer] を選択します。
2. [Step 1: Configure Load Balancer] で、以下の操作を行います。

   a. [Name] (名前) に、対象のロードバランサーを表現した説明的な名前を入力します。
   b. [Scheme] (スキーム) で、[internet-facing] (インターネット向け) を選択します。
   c. [Load Balancer Protocol] (ロードバランサーのプロトコル) で [HTTPS] を選択し、[Load Balancer Port] (ロードバランサーのポート) に 8445 を入力します。
   d. [VPC] で、使用する VPC を選択し、次にその VPC のサブネットを全て選択します。
   e. [次へ] をクリックします。
3. [Step 2: Configure Security Settings] (ステップ 2: セキュリティ設定を行う) で以下を実行します。

a. [Certificate type] (証明書のタイプ) で、[Choose a certificate from ACM] (ACM から証明書を選択する) を選択します。

b. [Certificate name] (証明書名) で、前の段階でリクエストしたプライベート証明書を選択します。

c. [次へ] をクリックします。

4. [Step 3: Configure Security Groups] (ステップ 3: セキュリティグループを設定する) で、新しいセキュリティグループを作成するか、または、HTTPS とポート 8445 でエージェントとブローカー間のインバウンドトラフィックとアウトバウンドトラフィックが可能な既存のセキュリティグループを選択します。

   [次へ] をクリックします。

5. [Step 4: Configure Routing] (ステップ 4: ルーティングの設定) で、以下の操作を行います。

   a. [Target group] (ターゲットグループ) で、[New target group] (新しいターゲットグループ) を選択します。

   b. [名前] に、ターゲットグループの名前を入力します。

   c. [Target type] (ターゲットタイプ) で [Instance] (インスタンス) を選択します。

   d. [Protocol] (プロトコル) で [HTTPS] を選択します。[Port (ポート)] に「8445」と入力します。[Protocol version] (プロトコルバージョン) で [HTTP1] を選択します。

   e. ヘルスチェックの [Protocol] (プロトコル) で、[HTTPS] を選択し、[Path] (パス) に `/health` を入力します。

   f. [次へ] をクリックします。

6. [Step 5: Register Targets] (ステップ 5: ターゲットを登録する) で、ブローカーインスタンスをすべて選択し、[Add to registered] (登録済みに追加) を選択します。[次へ: レビュー] を選択します。

7. [Create] (作成) を選択します。

ステップ 6: エージェントの起動

初期エージェントを作成し、ロードバランサーを使用するように設定して、エージェントから AMI を作成し、AMI を使用して残りのエージェントを起動します。これにより、すべてのエージェントが、同一のロードバランサー設定を使用するように設定されます。

エージェントの起動方法

1. NICE DCV サーバーを準備します。詳細については、「ステップ 1: NICE DCV サーバーを準備する」を参照してください。
2. ステップ 2: ロードバランサーの SSL 証明書を準備する で作成した CA 公開鍵のコピーを配置します。すべてのユーザーが読み取り可能なディレクトリを選択または作成します。CA 公開鍵ファイルもすべてのユーザーが読み取り可能である必要があります。
3. エージェントをインストールして設定します。エージェントのインストールおよび設定の詳細については、「ステップ 3: NICE DCV セッションマネージャーエージェントをセットアップする」を参照してください。

> **Important**
>
> エージェントの設定ファイルを作成して変更する場合
>
> - `broker_host` パラメータには、エージェントロードバランサーの DNS を入力します。
> - `ca_file` パラメータには、前のステップで作成した CA 公開鍵ファイルへのパスを入力します。

4. ブローカーを認証サーバーとして使用するように NICE DCV サーバーを設定します。詳細については、「ステップ 4: ブローカーが認証サーバーとして使用されるように NICE DCV サーバーを設定する」を参照してください。

> **Important**
>
> NICE DCV サーバー設定ファイルに変更を加える場合
>
> - `ca-file` パラメータには、前のステップで使用したものと同じ CA 公開鍵ファイルパスを入力します。
> - `auth-token-verifier` パラメータには、*`broker_ip_or_dns`* のエージェントロードバランサーの DNS を使用します。

5. エージェントインスタンスを停止します。
6. 停止したエージェントインスタンスから AMI を作成します。詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」の「インスタンスからの Linux AMI の作成」を参照してください。

7. AMI を使用して残りのエージェントを起動し、作成したインスタンスプロファイルをそれらすべてのエージェントに割り当てます。
8. エージェントからロードバランサーへのネットワークトラフィックを許可するセキュリティグループをすべてのエージェントインスタンスに割り当てます。ネットワークポートの詳細については、「エージェント設定ファイル」を参照してください。

NICE DCV サーバーをターゲットにするためのタグの使用

カスタムタグをセッションマネージャーエージェントに割り当てて、カスタムタグと、それらが関連付けられている NICE DCV サーバーを識別して分類することができます。新しい NICE DCV セッションを作成する場合、それぞれのエージェントに割り当てられたタグに基づいて、NICE DCV サーバーのグループをターゲットにすることができます。エージェントタグに基づいて NICE DCV サーバーをターゲットにする方法については、「セッションマネージャーデベロッパーガイド」の「CreateSessionRequests」を参照してください。

タグはタグのキーと値のペアで構成され、ユースケースや環境に適した情報ペアを使用できます。ホストのハードウェア設定に基づいて、エージェントへのタグ付けを選択できます。例えば、ホストに 4 GB のメモリが搭載されているすべてのエージェントに `ram=4GB` というタグを付けることができます。または、目的に応じてエージェントにタグを付けることもできます。例えば、本稼働用ホストで実行されているすべてのエージェントに `purpose=production` というタグを付けることができます。

エージェントにタグを割り当てる方法

1. 希望するテキストエディタを使用して新しいファイルを作成し、`agent_tags.toml` などの説明的な名前を付けます。ファイルタイプは `.toml` とし、ファイル内容は TOML ファイル形式で指定する必要があります。
2. ファイル内で、タグのキーと値の新しい各ペアを `key=value` 形式を使用して新しい行に追加します。例:

```
tag1="abc"
tag2="xyz"
```

3. エージェント設定ファイルを開きます (Linux の場合は `/etc/dcv-session-manager-agent/agent.conf`、Windows の場合は `C:\Program Files\NICE \DCVSessionManagerAgent\conf\agent.conf`)。`tags_folder` で、タグファイルがあるディレクトリへのパスを指定します。

ディレクトリに複数のタグファイルが含まれている場合は、ファイル間で定義されたすべてのタグによってエージェントが適用されます。ファイルはアルファベット順に読み取られます。キーが同じタグが複数のファイルに含まれている場合、その値は、最後に読み取られたファイルの値で上書きされます。

4. ファイルを保存して閉じます。
5. エージェントを停止して再起動します。

   - Windows

     ```
     C:\> sc stop DcvSessionManagerAgentService
     ```

     ```
     C:\> sc start DcvSessionManagerAgentService
     ```

   - Linux

     ```
     $ sudo systemctl stop dcv-session-manager-agent
     ```

     ```
     $ sudo systemctl start dcv-session-manager-agent
     ```

外部認可サーバーの設定

認可サーバーとは、クライアント SDK とエージェントの認証および認可を行うサーバーです。

セッションマネージャーでは、デフォルトで、ブローカーを認可サーバーとして使用して、クライアント SDK の OAuth 2.0 アクセストークンと、エージェント用のソフトウェアステートメントを生成します。ブローカーを認可サーバーとして使用する場合、追加の設定は必要ありません。

外部認可サーバーとしてブローカーではなく Amazon Cognito を使用するようにセッションマネージャーを設定することができます。Amazon Cognito の詳細については、「Amazon Cognito デベロッパーガイド」を参照してください。

Amazon Cognito を認可サーバーとして使用するには

1. Amazon Cognito ユーザープールを作成します。詳細については、「Amazon Cognito デベロッパーガイド」の「Amazon Cognito ユーザープール」を参照してください。

   create-user-pool コマンドを入力し、プール名と、プールを作成するリージョンを指定します。

この例では、プールに dcv-session-manager-client-app という名前を付けて、us-east-1 でそれを作成します。

```
$ aws cognito-idp create-user-pool --pool-name dcv-session-manager-client-app --
region us-east-1
```

出力例

```
{
    "UserPoolClient": {
        "UserPoolId": "us-east-1_QLEXAMPLE",
        "ClientName": "dcv-session-manager-client-app",
        "ClientId": "15hhd8jij74hf32f24uEXAMPLE",
        "LastModifiedDate": 1602510048.054,
        "CreationDate": 1602510048.054,
        "RefreshTokenValidity": 30,
        "AllowedOAuthFlowsUserPoolClient": false
    }
}
```

次のステップで必要になるため、userPoolId を書きとめておきます。

2. ユーザープールの新しいドメインを作成します。create-user-pool-domain コマンドを使用して、前のステップで作成したユーザープールのドメイン名と userPoolId を指定します。

   この例では、ドメイン名を mydomain-544fa30f-c0e5-4a02-8d2a-a3761EXAMPLE とし、us-east-1 でそれを作成します。

   ```
   $ aws cognito-idp create-user-pool-domain --domain mydomain-544fa30f-
   c0e5-4a02-8d2a-a3761EXAMPLE --user-pool-id us-east-1_QLEXAMPLE --region us-east-1
   ```

   出力例

   ```
   {
       "DomainDescription": {
           "UserPoolId": "us-east-1_QLEXAMPLE",
           "AWSAccountId": "123456789012",
           "Domain": "mydomain-544fa30f-c0e5-4a02-8d2a-a3761EXAMPLE",
           "S3Bucket": "aws-cognito-prod-pdx-assets",
           "CloudFrontDistribution": "dpp0gtexample.cloudfront.net",
   ```

```
        "Version": "20201012133715",
        "Status": "ACTIVE",
        "CustomDomainConfig": {}
    }
}
```

ユーザープールドメインの形式は https://*domain_name*.auth.*region*.amazoncognito.com です。この例では、ユーザープールのドメイン名は https://mydomain-544fa30f-c0e5-4a02-8d2a-a3761EXAMPLE.auth.us-east-1.amazoncognito.com です。

3. ユーザープールのクライアントを作成します。create-user-pool-client コマンドを使用して、作成したユーザープールの userPoolId、クライアント用の名前、これを作成するリージョンを指定します。さらに、作成するユーザープールクライアントのシークレットを生成するかどうかを指定する --generate-secret オプションを含めます。

   この場合、クライアント名は dcv-session-manager-client-app で、us-east-1 リージョンでそれを作成します。

```
$ aws cognito-idp create-user-pool-client --user-pool-id us-east-1_QLEXAMPLE --
client-name dcv-session-manager-client-app --generate-secret --region us-east-1
```

出力例

```
{
    "UserPoolClient": {
        "UserPoolId": "us-east-1_QLEXAMPLE",
        "ClientName": "dcv-session-manager-client-app",
        "ClientId": "2l9273hp6k2ut5cugg9EXAMPLE",
        "ClientSecret": "1vp5e8nec7cbf4m9me55mbmht91u61hlh0a78rq1qki1lEXAMPLE",
        "LastModifiedDate": 1602510291.498,
        "CreationDate": 1602510291.498,
        "RefreshTokenValidity": 30,
        "AllowedOAuthFlowsUserPoolClient": false
    }
}
```

> **Note**
>
> ClientId と ClientSecret を書きとめておきます。デベロッパーが API リクエストのアクセストークンをリクエストしたときに、この情報の提供が必要になります。

4. ユーザープール用の新しい OAuth 2.0 リソースサーバーを作成します。リソースサーバーは、アクセス保護されたリソース用のサーバーです。アクセストークンの認証リクエストが処理されます。

   create-resource-server コマンドを使用して、ユーザープールの userPoolId、リソースサーバーの一意の識別子と名前、スコープ、ユーザープールを作成するリージョンを指定します。

   この例では、dcv-session-manager を識別子と名前として使用し、sm_scope をスコープの名前と説明として使用します。

   ```
   $ aws cognito-idp create-resource-server --user-pool-id us-east-1_QLEXAMPLE
    --identifier dcv-session-manager --name dcv-session-manager --scopes
    ScopeName=sm_scope,ScopeDescription=sm_scope --region us-east-1
   ```

   出力例

   ```
   {
       "ResourceServer": {
           "UserPoolId": "us-east-1_QLEXAMPLE",
           "Identifier": "dcv-session-manager",
           "Name": "dcv-session-manager",
           "Scopes": [
           {
               "ScopeName": "sm_scope",
               "ScopeDescription": "sm_scope"
           }]
       }
   }
   ```

5. ユーザープールのクライアントを更新します。

   update-user-pool-client コマンドを使用します。ユーザープールの userPoolId、ユーザープールのクライアントの ClientId、リージョンを指定します。--allowed-o-auth-flows の場合、クライアント ID とクライアントシークレットを併用することで、クライアントがトークンエンドポイントからアクセストークンを取得することを示すために client_credentials を

指定します。--allowed-o-auth-scopes の場合、リソースサーバー識別子とスコープ名として *resource_server_identifier*/*scope_name* を指定します。クライアントが Cognito ユーザープールとの通信時に OAuth プロトコルに従うことを許可されていることを示すために --allowed-o-auth-flows-user-pool-client を含めます。

```
$ aws cognito-idp update-user-pool-client --user-pool-id us-east-1_QLEXAMPLE --
client-id 2l9273hp6k2ut5cugg9EXAMPLE --allowed-o-auth-flows client_credentials --
allowed-o-auth-scopes dcv-session-manager/sm_scope --allowed-o-auth-flows-user-
pool-client --region us-east-1
```

出力例

```
{
    "UserPoolClient": {
        "UserPoolId": "us-east-1_QLEXAMPLE",
        "ClientName": "dcv-session-manager-client-app",
        "ClientId": "2l9273hp6k2ut5cugg9EXAMPLE",
        "ClientSecret": "1vp5e8nec7cbf4m9me55mbmht91u61hlh0a78rq1qki1lEXAMPLE",
        "LastModifiedDate": 1602512103.099,
        "CreationDate": 1602510291.498,
        "RefreshTokenValidity": 30,
        "AllowedOAuthFlows": [
            "client_credentials"
        ],
        "AllowedOAuthScopes": [
            "dcv-session-manager/sm_scope"
        ],
        "AllowedOAuthFlowsUserPoolClient": true
    }
}
```

> **Note**
>
> これで、ユーザープールによるアクセストークンの提供と認証を行う準備が整いました。この例では、認可サーバーの URL は https://cognito-idp.*us-east-1*.amazonaws.com/*us-east-1_QLEXAMPLE*/.well-known/jwks.json です。

6. 設定をテストします。

```
$ curl -H "Authorization: Basic `echo -
n 2l9273hp6k2ut5cugg9EXAMPLE:1vp5e8nec7cbf4m9me55mbmht91u61hlh0a78rq1qki1lEXAMPLE
 | base64`" -H "Content-Type: application/x-www-form-urlencoded" -X
 POST "https://mydomain-544fa30f-c0e5-4a02-8d2a-a3761EXAMPLE.auth.us-
east-1.amazoncognito.com/oauth2/token?grant_type=client_credentials&scope=dcv-
session-manager/sm_scope"
```

出力例

```
{
"access_token":"eyJraWQiOiJGQ0VaRFpJUUptT3NSaW41MmtqaDdEbTZYb0RnSTQ5b2VUT0cxUUI1Q2VJPSIsImF
Zkfi0HIDsd6audjTXKzHlZGScr6ROdZtId5dThkpEZiSx0YwiiWe9crAlqoazlDcCsUJHIXDtgKW64pSj3-
uQQGg1Jv_tyVjhrA4JbD0k67WS2V9NW-
uZ7t4zwwaUmOi3KzpBMi54fpVgPaewiVlUm_aS4LUFcWT6hVJjiZF7om7984qb2gOa14iZxpXPBJTZX_gtG9EtvnS9u
"expires_in":3600,
"token_type":"Bearer"
}
```

7. register-auth-server コマンドを使用して、ブローカーに使用する外部認可サーバーを登録します。

```
$ sudo -u root dcv-session-manager-broker register-auth-server --url  https://
cognito-idp.us-east-1.amazonaws.com/us-east-1_QLEXAMPLE/.well-known/jwks.json
```

これで、デベロッパーがサーバーを使用してアクセストークンをリクエストできるようになりました。アクセストークンをリクエストするときは、ここで生成されたクライアント ID、クライアントシークレット、およびサーバー URL を提供します。アクセストークンのリクエストの詳細については、「NICE DCV セッションマネージャーデベロッパーガイド」の「アクセストークンの取得と API リクエストの提出」を参照してください。

ブローカー永続性の設定

セッションマネージャーのブローカーは、外部データベースとの統合に対応しています。外部データベースを使用すると、セッションマネージャーでステータスデータとキーを後で利用できるように永続化することができます。実際、ブローカーデータはクラスターに分散されるため、ホストの再起動が必要な場合やクラスターが終了した場合に、データ損失の影響を受けやすくなります。この機能を有効にすると、ブローカーノードの追加と削除が可能になります。また、クラスターの停止や再起動

を行っても、キーを再生成する必要もなく、NICE DCV サーバーの開閉に関する情報が失われることもありません。

以下のタイプの情報が永続化されるように設定することができます。

- クライアントとの接続を確立するためのセッション設定用キー
- インフライトセッションデータ
- NICE DCV サーバーステータス

NICE DCV セッションマネージャーでは、DynamoDB、MariaDB、MySQL のデータベースがサポートされています。この機能を使用するには、これらのデータベースの 1 つをセットアップして管理する必要があります。ブローカーマシンが Amazon EC2 でホストされている場合は、追加のセットアップが不要であるため、外部データベースとして DynamoDB を使用することをお勧めします。

> **Note**
>
> 外部データベースを実行すると追加コストが発生する可能性があります。DynamoDB 料金の詳細については、「プロビジョンドキャパシティーの料金」を参照してください。

DynamoDB で永続化されるようにブローカーを設定する

DynamoDB へのデータ保存が開始されるようにブローカーを設定します。

1. 任意のテキストエディタを使用して `/etc/dcv-session-manager-broker/session-manager-broker.properties` を開き、以下を実行します。
   - `enable-persistence = true` を設定する
   - `persistence-db = dynamodb` を設定する
   - `dynamodb-region` の場合、ブローカーデータが含まれているテーブルを保存する &aws;リージョンを指定します。サポートされているリージョンのリストについては、「DynamoDB サービスエンドポイント」を参照してください。
   - `dynamodb-table-rcu` の場合、各テーブルでサポートされている読み取りキャパシティーユニット (RCU) の量を指定します。RCU の詳細については、「DynamoDB プロビジョンドキャパシティー」を参照してください。

- dynamodb-table-wcu の場合、各テーブルでサポートされている書き込みキャパシティーユニット (WCU) の量を指定します。WCU の詳細については、「DynamoDB プロビジョンドキャパシティー」を参照してください。
- dynamodb-table-name-prefix の場合、各 DynamoDB テーブルに追加されるプレフィックスを指定します (同じアカウントを使用して複数のブローカークラスターを識別する際に便利です)。英数字、ドット、ダッシュ、下線のみを使用できます。

2. クラスター内のすべてのブローカーを停止します。ブローカーごとに次のコマンドを実行します。

```
sudo systemctl stop dcv-session-manager-broker
```

3. クラスター内のすべてのブローカーが停止していることを確認した上で、すべてのブローカーを再起動します。次のコマンドを実行して各ブローカーを開始します。

```
sudo systemctl start dcv-session-manager-broker
```

ブローカーのホストには、DynamoDB API を呼び出すためのアクセス許可が必要です。Amazon EC2 インスタンスでは、認証情報は、Amazon EC2 メタデータサービスを使用して自動的に取得されます。異なる認証情報を指定する必要がある場合は、サポートされている認証情報取得方法 (Java システムプロパティや環境変数など) のいずれかを使用してそれらを設定することができます。詳細については、「認証情報の提供と取得」を参照してください。

MariaDB/MySQL で永続化させるためにブローカーを設定する

> **Note**
>
> /etc/dcv-session-manager-broker/session-manager-broker.properties ファイルには機密データが含まれています。デフォルトでは、書き込みアクセスはルートに制限され、読み取りアクセスはルートおよびブローカーを実行しているユーザーに制限されます。デフォルトでは、これは dcvsmbroker ユーザーです。ブローカーのスタートアップ時に、期待される許可がファイルに含まれていることが確認されます。

MariaDB/MySQL でデータの永続化を開始するためにブローカーを設定します。

1. 任意のテキストエディタで /etc/dcv-session-manager-broker/session-manager-broker.properties を開き、以下の編集を行います。

- `enable-persistence = true` を設定する
- `persistence-db = mysql` を設定する
- `jdbc-connection-url = jdbc:mysql://<db_endpoint>:<db_port>/<db_name>?createDatabaseIfNotExist=true` を設定する

  この設定では、<db_endpoint> はデータベースエンドポイント、<db_port> はデータベースポート、<db_name> はデータベース名です。
- `jdbc-user` について、データベースにアクセスできるユーザーの名前を指定します。
- `jdbc-password` について、データベースにアクセスできるユーザーのパスワードを指定します。

2. クラスター内のすべてのブローカーを停止します。ブローカーごとに次のコマンドを実行します。

```
sudo systemctl stop dcv-session-manager-broker
```

3. クラスター内のすべてのブローカーが停止していることを確認した上で、すべてのブローカーを再起動します。ブローカーごとに次のコマンドを実行します。

```
sudo systemctl start dcv-session-manager-broker
```

NICE DCV 接続ゲートウェイとの統合

NICE DCV 接続ゲートウェイは、ユーザーが LAN または VPC への単一のアクセスポイントを通じて NICE DCV サーバーフリートにアクセスできるようにする、インストール可能なソフトウェアパッケージです。

インフラストラクチャに NICE DCV 接続ゲートウェイ経由でアクセス可能な NICE DCV サーバーが含まれている場合は、NICE DCV 接続ゲートウェイを統合するようにセッションマネージャーを構成できます。以下のセクションで説明する手順を実行すると、ブローカーは接続ゲートウェイのセッションリゾルバーとして機能します。つまり、ブローカーは追加の HTTP エンドポイントを公開します。接続ゲートウェイはエンドポイントに対して API 呼び出しを行い、ブローカーが選択したホストに NICE DCV 接続をルーティングするために必要な情報を取得します。

セッションマネージャーブローカーを NICE DCV 接続ゲートウェイのセッションリゾルバーとして設定

セッションマネージャーブローカー側

1. 任意のテキストエディタを使用して /etc/dcv-session-manager-broker/session-manager-broker.properties を開き、以下の変更を適用します。
   - enable-gateway = true を設定する
   - gateway-to-broker-connector-https-port を空いている TCP ポートに設定する (デフォルトは 8447)
   - gateway-to-broker-connector-bind-host をNICE DCV ゲートウェイ接続のためにブローカーによりバインドされるホストの IP アドレスに設定する (デフォルトは 0.0.0.0)
2. 次に、以下のコマンドを実行して Broker を停止し、再起動します。

   ```
   sudo systemctl stop dcv-session-manager-broker
   ```

   ```
   sudo systemctl start dcv-session-manager-broker
   ```
3. ブローカーの自己署名証明書のコピーを取得し、ユーザーディレクトリに入れます。

   ```
   sudo cp /var/lib/dcvsmbroker/security/dcvsmbroker_ca.pem $HOME
   ```

   これは、次のステップでNICE DCV 接続ゲートウェイをインストールするときに必要になります。

NICE DCV 接続ゲートウェイ側

- NICE DCV 接続ゲートウェイのドキュメントのセクションに従ってください。

  NICE DCV 接続ゲートウェイはブローカーに対して HTTP API 呼び出しを行うため、ブローカーが自己署名証明書を使用している場合はブローカーの証明書を NICE DCV 接続ゲートウェイのホスト (前のステップで取得) にコピーし、NICE DCV 接続ゲートウェイ構成の [resolver] セクションでパラメーターを ca-file に設定する必要があります。

オプション - TLS クライアント認証の有効化

前のステップを完了すると、セッションマネージャーと接続ゲートウェイは安全なチャネルを介して通信できるようになり、接続ゲートウェイはセッションマネージャブローカーの識別情報を確認できます。安全なチャネルを確立する前にセッションマネージャブローカーにも接続ゲートウェイの識別情報を確認させる必要がある場合は、次のセクションの手順に従って TLS クライアント認証機能を有効にする必要があります。

> **Note**
>
> セッションマネージャーがロードバランサーの背後にある場合、Application Load Balancer (ALB) やゲートウェイロードバランサー (GLB) などの TLS 接続終了機能のあるロードバランサーでは TLS クライアント認証を有効にできません。サポートされるのは、ネットワークロードバランサー (NLB) などの TLS 終了機能を搭載していないロードバランサーのみです。ALB または GLB を使用する場合は、特定のセキュリティグループのみがロードバランサーに接続できるように設定して、セキュリティレベルを高めることができます。セキュリティグループの詳細については、「VPC のセキュリティグループ」をご覧ください。

セッションマネージャーブローカー側

1. セッションマネージャーブローカーと NICE DCV 接続ゲートウェイ間の通信で TLS クライアント認証を有効にするには、次の手順に従ってください。
2. 以下のコマンドを実行して、必要なキーと証明書を生成します。コマンドの出力には、認証情報が生成されたフォルダと、TrustStore ファイルの作成に使用されたパスワードが表示されます。

```
sudo /usr/share/dcv-session-manager-broker/bin/gen-gateway-certificates.sh
```

3. NICE DCV 接続ゲートウェイのプライベートキーと自己署名証明書のコピーをユーザーディレクトリに入れます。これは、次のステップで、NICE DCV 接続ゲートウェイで TLS クライアント認証を有効にするときに必要になります。

```
sudo cp /etc/dcv-session-manager-broker/resolver-creds/dcv_gateway_key.pem $HOME
```

```
sudo cp /etc/dcv-session-manager-broker/resolver-creds/dcv_gateway_cert.pem $HOME
```

4. 次に、任意のテキストエディタを使用して /etc/dcv-session-manager-broker/session-manager-broker.properties を開き、以下を実行します。

- `enable-tls-client-auth-gateway` を `true` に設定します。
- `gateway-to-broker-connector-trust-store-file` を前のステップで作成した TrustStore ファイルのパスに設定します
- `gateway-to-broker-connector-trust-store-pass` を前のステップで TrustStore ファイルの作成に使用したパスワードに設定します。

5. 次に、以下のコマンドを実行して Broker を停止し、再起動します。

```
sudo systemctl stop dcv-session-manager-broker
```

```
sudo systemctl start dcv-session-manager-broker
```

NICE DCV 接続ゲートウェイ側

- NICE DCV 接続ゲートウェイのドキュメントのセクションに従ってください。
  - `[resolver]` セクションで `cert-file` パラメータを設定するときは、前のステップでコピーした証明書ファイルのフルパスを使用します
  - `[resolver]` セクションで `cert-key-file` パラメーターを設定するときは、前のステップでコピーしたキーファイルのフルパスを使用します

NICE DCV セッションマネージャー NICE DCV サーバー - DNS マッピング

NICE DCV 接続ゲートウェイは、DCV サーバーインスタンスに接続するために NICE DCV サーバーの DNS 名を必要とします。このセクションでは、各 DCV サーバーと、関連する DNS 名とのマッピングを含む JSON ファイルを定義する方法を説明します。

ファイル構造

マッピングは、次のフィールドを持つ JSON オブジェクトのリストで構成されます。

```
[
 {
  “ServerIdType”: "Ip",
  “ServerId”: "192.168.0.1",
  “DnsNames”:
  {
```

```
   “InternalDnsName”: "internal"
  }
 },
 ...
]
```

実行する条件は以下のとおりです。

ServerIdType:

値がどのタイプの ID を参照しているかを示します。現在使用可能な値は IPAddress、AgentServerID、および instanceId です。

Ip:

Amazon EC2 とオンプレミスの両方のインフラストラクチャで使用できます。システム管理者は ifconfig (Linux) または ipconfig (Windows) コマンドを使用してすばやく取得できます。この情報は DescribeServers API レスポンスでも確認できます。

Id:

Amazon EC2 とオンプレミスの両方のインフラストラクチャで使用できます。セッションマネージャーエージェントは、ホスト名または IP アドレスが変更されるたびに新しい UUID を作成します。この情報は DescribeServers API レスポンスで確認できます。

Host.Aws.Ec2InstanceId:

Amazon EC2 インスタンスでのみ使用でき、マシンを一意に識別します。インスタンスの再起動後も変更されません。http://169.254.169.254/latest/meta-data/instance-id に問い合わせることで、ホスト上で取得できます。この情報は DescribeServers API レスポンスでも確認できます。

ServerId:

ネットワーク内の各 NICE DCV サーバーを一意に識別する、指定されたタイプの ID。

DnsNames:

NICE DCV サーバーに関連付けられた DNS 名を含むオブジェクトです。このオブジェクトには以下が含まれます。

InternalDnsNames:

NICE DCV 接続ゲートウェイがインスタンスへの接続に使用する DNS 名。

ファイルからマッピングをロードするにはセッションマネージャーブローカー CLI コマンド `register-server-dns-mapping` (コマンドページリファレンス: register-server-dns-mapping) 、セッションマネージャーブローカーに現在ロードされているマッピングを一覧表示するには `describe-server-dns-mappings` を使用してください (コマンドページリファレンス: describe-server-dns-mappings)。

永続的

複数のブローカーまたはクラスター全体がダウンした場合にマッピングが失われないように、セッションマネージャーブローカーの永続化機能を有効にすることを強くお勧めします。データ永続化の有効化について詳しくは、「ブローカーの永続化の設定」を参照してください。

Amazon CloudWatch との統合

セッションマネージャーでは、Amazon EC2 インスタンスで実行されているブローカー用 Amazon CloudWatch と、オンプレミスのホストで実行されているブローカーとの統合がサポートされています。

Amazon CloudWatch は、Amazon Web Services (AWS) リソースと、AWS で実行されているアプリケーションをリアルタイムでモニターリングします。CloudWatch を使用してメトリクスを収集し、追跡できます。メトリクスとは、リソースやアプリケーションに関して測定できる変数です。詳細については、Amazon CloudWatch ユーザーガイドを参照してください。

次のメトリクスデータが Amazon CloudWatch に送信されるように、セッションマネージャーブローカーを設定することができます。

- `Number of DCV servers` — ブローカーによって管理されている DCV サーバーの数。
- `READY` — ブローカーによって管理されており `Number of ready DCV servers` 状態にある DCV サーバーの数。
- `Number of DCV sessions` — ブローカーによって管理されている DCV セッションの数。
- `Number of DCV console sessions` — ブローカーによって管理されている DCV コンソールセッションの数。
- `Number of DCV virtual sessions` — ブローカーによって管理されている DCV 仮想セッションの数。
- `Heap memory used` — ブローカーによって使用されているヒープメモリの量。
- `Off-heap memory used` — ブローカーによって使用されるオフヒープメモリの量。

- `Describe sessions request time` — DescribeSessions API リクエストの完了に要した時間。
- `Delete sessions request time` — DeleteSessions API リクエストの完了に要した時間。
- `Create sessions request time` — CreateSessions API リクエストの完了に要した時間。
- `Get session connection data request time` — GetSessionConnectionData API リクエストの完了に要した時間。
- `Update session permissions sequest time` — UpdateSessionPermissions API リクエストの完了に要した時間。

Amazon CloudWatch にメトリクスデータが送信されるようにブローカーを設定する方法

1. 任意のテキストエディタを使用して `/etc/dcv-session-manager-broker/session-manager-broker.properties` ファイルを開き、以下を実行します。
   - `enable-cloud-watch-metrics` を `true` に設定します。
   - `cloud-watch-region` の場合、メトリクスデータを収集するリージョンを指定します。

   > **Note**
   >
   > ブローカーが Amazon EC2 インスタンスで実行されている場合、このパラメータはオプションです。このリージョンは、インスタンスメタデータサービス (IMDS) から自動的に取得されます。ブローカーをオンプレミスのホストで実行している場合、このパラメータは必須です。

2. ブローカーを停止して再起動します。

```
$  sudo systemctl stop dcv-session-manager-broker
```

```
$  sudo systemctl start dcv-session-manager-broker
```

ブローカーホストには `cloudwatch:PutMetricData` API を呼び出すアクセス許可も必要です。AWS 認証情報は、サポートされている認証情報取得方法のいずれかを使用して取得することができます。詳細については、「AWS 認証情報の提供と取得」を参照してください。

NICE DCV セッションマネージャーのアップグレード

次のトピックでは、セッションマネージャーをアップグレードする方法について説明します。

> **Note**
>
> セッションマネージャーブローカーをアップグレードする前に、すべてのセッションマネージャーエージェントをアップグレードすることを強くお勧めします。これは、新機能導入の際に互換性の問題が発生するのを避けるためです。

トピック

- NICE DCV セッションマネージャーエージェントのアップグレード
- NICE DCV セッションマネージャーブローカーのアップグレード

NICE DCV セッションマネージャーエージェントのアップグレード

Linux host

> **Note**
>
> 以下は、64 ビット x86 ホストにエージェントをインストールする手順です。64 ビット ARM ホストにエージェントのインストールについて、Amazon Linux、RHEL、Centos の場合は *x86_64*が `aarch64` になり、Ubuntu の場合は *amd64* が `arm64` になります。

Linux ホストでエージェントを更新するには

1. 次のコマンドを実行してエージェントを停止します。

   ```
   $  sudo systemctl stop dcv-session-manager-agent
   ```

2. インストールパッケージをダウンロードします。

   - Amazon Linux 2、RHEL 7.x、CentOS 7.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.el7.x86_64.rpm
```

- RHEL 8.x、CentOS Stream 8、Rocky Linux 8.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.el8.x86_64.rpm
```

- Ubuntu 20.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent_2023.1.732-1_amd64.ubuntu2004.deb
```

- SUSE Linux Enterprise 12

```
$  curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$  curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerAgents/
nice-dcv-session-manager-agent-2023.1.732-1.sles15.x86_64.rpm
```

3. パッケージをインストールします。

- Amazon Linux 2、RHEL 7.x、CentOS 7.x

```
$  sudo yum install -y nice-dcv-session-manager-
agent-2023.1.732-1.el7.x86_64.rpm
```

- RHEL 8.x、CentOS Stream 8、Rocky Linux 8.x

```
$  sudo yum install -y nice-dcv-session-manager-
agent-2023.1.732-1.el8.x86_64.rpm
```

- Ubuntu 20.04

```
$  sudo apt install ./nice-dcv-session-manager-
agent_2023.1.732-1_amd64.ubuntu2004.deb
```

- SUSE Linux Enterprise 12

```
$  sudo zypper install nice-dcv-session-manager-
agent-2023.1.732-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$  sudo zypper install nice-dcv-session-manager-
agent-2023.1.732-1.sles15.x86_64.rpm
```

4. エージェントを開始するには、次のコマンドを実行します。

```
$  sudo systemctl start dcv-session-manager-agent
```

Windows host

Windows ホストでエージェントを更新するには

1. エージェントのサービスを停止します。コマンドプロンプトで、次のコマンドを入力します。

```
C:\>  sc start DcvSessionManagerAgentService
```

2. エージェントインストーラーをダウンロードします。
3. インストーラーを実行します。ようこそ画面で、[Next] を選択します。
4. [EULA] 画面で、使用許諾書をしっかりと読んだ上で、条件に同意する場合は [I accept the terms] (同意します) を選択し、[Next] (次へ) を選択します。
5. インストールを開始するには [Install] (インストール) を選択します。
6. エージェントのサービスを再起動します。コマンドプロンプトで、次のコマンドを入力します。

```
C:\>  sc stop DcvSessionManagerAgentService
```

NICE DCV セッションマネージャーブローカーのアップグレード

ブローカーをアップグレードするには

1. ブローカーをアップグレードするホストに接続します。

2. ブローカーのサービスを停止します。

```
$  sudo systemctl stop dcv-session-manager-broker
```

3. インストールパッケージをダウンロードします。

   - Amazon Linux 2、RHEL 7.x、CentOS 7.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-
dcv-session-manager-broker-2023.1.410-1.el7.noarch.rpm
```

   - RHEL 8.x、CentOS Stream 8、Rocky Linux 8.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-
dcv-session-manager-broker-2023.1.410-1.el8.noarch.rpm
```

   - Ubuntu 20.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2023.1/SessionManagerBrokers/nice-
dcv-session-manager-broker-2023.1.410-1_all.ubuntu2004.deb
```

4. パッケージをインストールします。

   - Amazon Linux 2、RHEL 7.x、CentOS 7.x

```
$  sudo yum install -y nice-dcv-session-manager-
broker-2023.1.410-1.el7.noarch.rpm
```

   - RHEL 8.x、CentOS Stream 8、Rocky Linux 8.x

```
$  sudo yum install -y nice-dcv-session-manager-
broker-2023.1.410-1.el8.noarch.rpm
```

   - Ubuntu 20.04

```
$  sudo apt install -y nice-dcv-session-manager-
broker-2023.1.410-1_all.ubuntu2004.deb
```

5. ブローカーサービスを開始し、インスタンスを起動するたびにブローカーサービスが自動的に起動することを確認します。

```
$  sudo systemctl start dcv-session-manager-broker && sudo systemctl enable dcv-
session-manager-broker
```

ブローカー CLI リファレンス

このセクションでは、ブローカーのコマンドラインインターフェイス (CLI) コマンドを使用する方法について説明します。

外部認証サーバーを使用して OAuth 2.0 アクセストークンを生成する場合は、次のコマンドを使用します。

- register-auth-server
- list-auth-servers
- unregister-auth-server

OAuth 2.0 認証サーバーとしてセッションマネージャーブローカーを使用する場合は、次のコマンドを使用します。

- register-api-client
- describe-api-clients
- unregister-api-client
- renew-auth-server-api-key

次のコマンドを使用して、セクションマネージャーエージェントを管理します。

- generate-software-statement
- describe-software-statements
- deactivate-software-statement
- describe-agent-clients
- unregister-agent-client

DCV サーバーと DNS 名のマッピングファイルを管理するには、次のコマンドを使用します。

- register-server-dns-mappings
- describe-server-dns-mappings

register-auth-server

ブローカーで使用する外部認証サーバーを登録します。

デフォルトでは、セッションマネージャーはブローカーを認証サーバーとして使用し、OAuth 2.0 アクセストークンを生成します。ブローカーを認証サーバーとして使用する場合、追加の設定は必要ありません。

ただし、アクティブディレクトリや Amazon Cognito などの外部認証サーバーの使用を選択した場合は、このコマンドを使用して外部認証サーバーを登録する必要があります。

トピック

- 構文
- オプション
- 例

構文

```
sudo -u root dcv-session-manager-broker register-auth-server --url  server_url.well-known/jwks.json
```

オプション

`--url`

使用する外部認証サーバーの URL。認証サーバーの URL に `.well-known/jwks.json` を付加する必要があります。

型: 文字列

必須: はい

例

次の例では、外部認証サーバーを `https://my-auth-server.com/` の URL に登録します。

コマンド

```
sudo -u root dcv-session-manager-broker register-auth-server --url https://my-auth-server.com/.well-known/jwks.json
```

出力

```
Jwk url registered.
```

list-auth-servers

登録されている外部認証サーバーを一覧表示します。

トピック

- 構文
- 出力
- 例

構文

```
sudo -u root dcv-session-manager-broker list-auth-servers
```

出力

Urls

登録された外部認証サーバーの URL。

例

次の例では、登録されているすべての外部認証サーバーを一覧表示します。

コマンド

```
sudo -u root dcv-session-manager-broker list-auth-servers
```

出力

```
Urls: [ "https://my-auth-server.com/.well-known/jwks.json" ]
```

unregister-auth-server

外部認証サーバーの登録を解除します。外部認証サーバーは、その登録を解除すると、OAuth 2.0 アクセストークンの生成に使用できなくなります。

トピック

- 構文
- オプション
- 出力
- 例

構文

```
sudo -u root dcv-session-manager-broker unregister-auth-server --url  server_url.well-known/jwks.json
```

オプション

`--url`

登録を解除する外部認証サーバーの URL。認証サーバーの URL に `.well-known/jwks.json` を付加する必要があります。

型: 文字列

必須: はい

出力

`Url`

登録を解除した外部認証サーバの URL。

例

次の例では、外部認証サーバーを `https://my-auth-server.com/` の URL に登録します。

コマンド

```
sudo -u root dcv-session-manager-broker unregister-auth-server --url https://my-auth-server.com/.well-known/jwks.json
```

出力

```
Jwk urlhttps://my-auth-server.com/.well-known/jwks.json unregistered
```

register-api-client

セッションマネージャークライアントをブローカーに登録してクライアント認証情報を生成します。この情報は、API リクエストの実行に必要な OAuth 2.0 アクセストークンを取得するためにクライアントで使用されます。

> **Important**
> 認証情報は必ず安全な場所に保存してください。後で復元することはできません。

このコマンドは、ブローカーが OAuth 2.0 認証サーバーとして使用される場合にのみ使用されます。

トピック

- 構文
- オプション
- 出力
- 例

構文

```
sudo -u root dcv-session-manager-broker register-api-client --client-name client_name
```

オプション

--name

セクションマネージャークライアントを識別するために使用される一意の名前。

型: 文字列

必須: はい

出力

client-id

OAuth 2.0 アクセストークンの取得のためにセッションマネージャークライアントにより使用される一意のクライアント ID。

client-password

OAuth 2.0 アクセストークンの取得のためにセッションマネージャークライアントにより使用されるパスワード。

例

次の例では、my-sm-client という名前のクライアントを登録します。

コマンド

```
sudo -u root dcv-session-manager-broker register-api-client --client-name my-sm-client
```

出力

```
client-id: 21cfe9cf-61d7-4c53-b1b6-cf248EXAMPLE
client-password: NjVmZDRlN2ItNjNmYS00M2QxLWFlZmMtZmNmMDNkMEXAMPLE
```

describe-api-clients

ブローカーに登録されているセッションマネージャークライアントを一覧表示します。

トピック

- 構文
- 出力
- 例

構文

```
sudo -u root dcv-session-manager-broker describe-api-clients
```

出力

name

セッションマネージャークライアントの一意の名前。

id

セッションマネージャークライアントの一意の ID。

active

セッションマネージャークライアントのステータスを示します。クライアントがアクティブな場合、値は `true` になり、それ以外の場合は `false` になります。

例

次の例では、登録されているセッションマネージャークライアントの一覧を示します。

コマンド

```
sudo -u root dcv-session-manager-broker describe-api-clients
```

出力

```
Api clients
[ {
"name" : "client-abc",
"id" : "f855b54b-40d4-4769-b792-b727bEXAMPLE",
"active" : false
}, {
"name" : "client-xyz",
```

```
"id" : "21cfe9cf-61d7-4c53-b1b6-cf248EXAMPLE",
"active" : true
}]
```

unregister-api-client

登録済みのセッションマネージャークライアントを非アクティブ化します。非アクティブ化されたセッションマネージャークライアントでは、その認証情報を使用して OAuth 2.0 アクセストークンを取得できなくなります。

トピック

- 構文
- オプション
- 例

構文

```
sudo -u root dcv-session-manager-broker unregister-api-client --client-id client_id
```

オプション

`--client -id`

非アクティブ化するセッションマネージャークライアントの ID。

型: 文字列

必須: はい

例

次の例では、クライアント ID が f855b54b-40d4-4769-b792-b727bEXAMPLE であるセッションマネージャークライアントを非アクティブ化します。

コマンド

```
sudo -u root dcv-session-manager-broker unregister-api-client --client-id
 f855b54b-40d4-4769-b792-b727bEXAMPLE
```

出力

```
Client f855b54b-40d4-4769-b792-b727bEXAMPLE unregistered.
```

renew-auth-server-api-key

セッションマネージャークライアントに提供される OAuth 2.0 アクセストークンに署名するために、ブローカーによって使用されるパブリックキーとプライベートキーを更新します。キーを更新する場合は、API リクエストの実行に新しいプライベートキーが必要になるため、新しいプライベートキーをデベロッパーに提供する必要があります。

トピック

- 構文
- 例

構文

```
sudo -u root dcv-session-manager-broker renew-auth-server-api-key
```

例

次の例では、パブリックキーとプライベートキーを更新します。

コマンド

```
sudo -u root dcv-session-manager-broker renew-auth-server-api-key
```

出力

```
Keys renewed.
```

generate-software-statement

ソフトウェアステートメントを生成します。

通信を有効にするには、エージェントをブローカーに登録する必要があります。エージェントをブローカーに登録するにはソフトウェアステートメントが必要です。エージェントでは、ソフトウェア

ステートメントを取得したら、OAuth 2.0 動的クライアント登録プロトコルを使用してブローカーに自動的に登録されます。エージェントがブローカーに登録されると、ブローカーとの認証に使用するクライアント ID とクライアントシークレットを受け取ります。

ブローカーとエージェントでは、初回インストール時にデフォルトのソフトウェアステートメントを受け取って使用します。デフォルトのソフトウェアステートメントを引き続き使用することも、新しいソフトウェアステートメントを生成することもできます。新しいソフトウェアステートメントを生成する場合は、そのソフトウェアステートメントをエージェントの新しいファイルに配置し、agent.conf ファイルの agent.software_statement_path パラメータにファイルパスを追加する必要があります。この作業が完了した後、エージェントを停止して再起動すると、新しいソフトウェアステートメントを使用してブローカーに登録できます。

トピック

- 構文
- 出力
- 例

構文

```
sudo -u root dcv-session-manager-broker generate-software-statement
```

出力

software-statement

ソフトウェアステートメント。

例

次の例では、ソフトウェアステートメントを生成します。

コマンド

```
sudo -u root dcv-session-manager-broker generate-software-statement
```

出力

```
software-statement:
 ewogICJpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiYWN0aXZlIiA6IHRydWUsCi
```

describe-software-statements

既存のソフトウェアステートメントについて説明します。

トピック

- 構文
- 出力
- 例

構文

```
sudo -u root dcv-session-manager-broker describe-software-statements
```

出力

`software-statement`

ソフトウェアステートメント。

`issued-at`

ソフトウェアが生成された日時。

`is-active`

ソフトウェアステートメントの現在の状態。ソフトウェアステートメントがアクティブな場合は `true`。それ以外の場合は `false`。

例

次の例では、ソフトウェアステートメントを生成します。

コマンド

```
sudo -u root dcv-session-manager-broker describe-software-statements
```

出力

```
Software Statements
[ {
"software-statement" :
 "ewogICJpZCIgOiAiYmEEXAMPLEYtNzUwNy00YmFhLTliZWItYTA1MmJjZTE3NDJjIiwKICAiaXNzdWVkQXQiIDogMTU5N
"issued-at" : "2020.08.05 15:38:32 +0000",
"is-active" : "true"
}, {
"software-statement" :
 "EXAMPLEpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiaXNzdWEXAMPLEDogMTU5N
"issued-at" : "2020.08.07 10:24:41 +0000",
"is-active" : "true"
} ]
```

deactivate-software-statement

ソフトウェアステートメントを非アクティブ化します。ソフトウェアステートメントは、非アクティブ化すると、エージェント登録に使用できなくなります。

トピック

- 構文
- オプション
- 例

構文

```
sudo -u root dcv-session-manager-broker deactivate-software-statement --software-
statement software_statement
```

オプション

`--software-statement`

非アクティブ化するソフトウェアステートメント。

型: 文字列

必須: はい

例

次の例では、ソフトウェアステートメントを非アクティブ化します。

コマンド

```
sudo -u root dcv-session-manager-broker deactivate-software-statement --software-
statement
 EXAMPLEpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiaXNEXAMPLEQiIDogMTU5Nj
```

出力

```
Software statement
 EXAMPLEpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiaXNEXAMPLEQiIDogMTU5Nj
 deactivated
```

describe-agent-clients

ブローカーに登録されているエージェントを記述します。

トピック

- 構文
- 出力
- 例

構文

```
sudo -u root dcv-session-manager-broker describe-agent-clients
```

出力

name

エージェントの名前。

id

エージェントの一意の ID。

active

エージェントの状態。エージェントがアクティブな場合は `true`。それ以外の場合は `false`。

例

次の例ではエージェントを記述します。

コマンド

```
sudo -u root dcv-session-manager-broker describe-agent-clients
```

出力

```
Session manager agent clients
[ {
"name" : "test",
"id" : "6bc05632-70cb-4410-9e54-eaf9bEXAMPLE",
"active" : true
}, {
"name" : "test",
"id" : "27131cc2-4c71-4157-a4ca-bde38EXAMPLE",
"active" : true
}, {
"name" : "test",
"id" : "308dd275-2b66-443f-95af-33f63EXAMPLE",
"active" : false
}, {
"name" : "test",
"id" : "ce412d1b-d75c-4510-a11b-9d9a3EXAMPLE",
"active" : true
} ]
```

unregister-agent-client

ブローカーへのエージェントの登録を解除します。

トピック

- 構文
- オプション

- 例

構文

```
sudo -u root dcv-session-manager-broker unregister-agent-client --client-id client_id
```

オプション

`--client-id`

登録を解除するエージェントの ID。

型: 文字列

必須: はい

例

次の例ではエージェントの登録を解除します。

コマンド

```
sudo -u root dcv-session-manager-broker unregister-agent-client --client-id
 3b0d7b1d-78c7-4e79-b2e1-b976dEXAMPLE
```

出力

```
Agent client 3b0d7b1d-78c7-4e79-b2e1-b976dEXAMPLE unregistered
```

register-server-dns-mappings

JSON ファイルに基づく DCV サーバーと DNS 名のマッピングを登録します。

構文

```
sudo -u root dcv-session-manager-broker register-server-dns-mappings --file-
path file_path
```

オプション

`--file-path`

DCV サーバーと DNS 名のマッピングを含むファイルのパスです。

型: 文字列

必須: はい

例

次の例は、/tmp/mappings.json ファイルから DCV サーバーと DNS 名のマッピングを登録します。

コマンド

```
sudo -u root dcv-session-manager-broker register-server-dns-mappings --file-path /tmp/mappings.json
```

出力

```
Successfully loaded 2 server id - dns name mappings from file /tmp/mappings.json
```

describe-server-dns-mappings

現在利用可能な DCV サーバー と DNS 名マッピングを示します。

構文

```
sudo -u root dcv-session-manager-broker describe-server-dns-mappings
```

出力

`serverIdType`

サーバー ID のタイプ

`serverId`

サーバーに付けられた一意の ID

dnsNames

内部 DNS 名と外部 DNS 名

internalDnsNames

内部 DNS 名

externalDnsNames

外部 DNS 名

例

次の例は、登録されている DCV サーバー と DNS 名のマッピングの一覧を示します。

コマンド

```
sudo -u root dcv-session-manager-broker describe-server-dns-mappings
```

出力

```
[
{
 "serverIdType" : "Id",
 "serverId" : "192.168.0.1",
 "dnsNames" : {
  "internalDnsName" : "internal1",
  "externalDnsName" : "external1"
 }
},
{
 "serverIdType" : "Host.Aws.Ec2InstanceId",
 "serverId" : "i-0648aee30bc78bdff",
 "dnsNames" : {
  "internalDnsName" : "internal2",
  "externalDnsName" : "external2"
 }
}
]
```

設定ファイルリファレンス

このセクションでは、エージェントとブローカーの設定ファイルについて説明します。

トピック

- ブローカー設定ファイル
- エージェント設定ファイル

ブローカー設定ファイル

ブローカー設定ファイル (/etc/dcv-session-manager-broker/session-manager-broker.properties) には、セッションマネージャー機能をカスタマイズするために設定できるパラメータが含まれています。希望するテキストエディタを使用して設定ファイルを手動で編集することができます。

> **Note**
>
> /etc/dcv-session-manager-broker/session-manager-broker.properties ファイルには機密データが含まれています。デフォルトでは、書き込みアクセスはルートに制限され、読み取りアクセスはルートおよびブローカーを実行しているユーザーに制限されます。デフォルトでは、これは dcvsmbroker ユーザーです。ブローカーのスタートアップ時に、期待される許可がファイルに含まれていることが確認されます。

下表にブローカー設定ファイルのパラメータを示します。

Parameter Name	必須	デフォルト値	説明
broker-java-home	No		ブローカーがシステムのデフォルトディレクトリの代わりに使用する Java ホームディレクトリのパスを指定します。設定すると、ブローカーは起動時に <broker-java-home>

Parameter Name	必須	デフォルト値	説明
			/bin/java を使用します。 ヒント: ブローカーには Java ランタイム環境 11 が必要です。インストールされていない場合は、インストールが成功すると依存関係としてインストールされます。バージョン 11 がデフォルトの Java 環境として設定されていない場合は、以下のコマンドを使用してホームディレクトリを取得できます。 `$ sudo alternatives --display java`
sessions-screenshot-max-width	No	160	GetSessionScreenshots API を使用して、セッションスクリーンショットの最大幅 (ピクセル単位) を指定します。
sessions-screenshot-max-height	No	100	GetSessionScreenshots API を使用して、セッションスクリーンショットの最大高さ (ピクセル単位) を指定します。

Parameter Name	必須	デフォルト値	説明
session-screenshot-format	No	png	GetSessionScreenshots API を使用して撮影したセッションスクリーンショットのイメージファイル形式です。
create-sessions-queue-max-size	No	1000	キューに入る可能性がある未処理の CreateSessions API リクエストの最大数です。キューがいっぱいになると、新たな未処理リクエストは拒否されます。
create-sessions-queue-max-time-seconds	No	1800	未処理の CreateSessions API リクエストがキューに残留できる最長時間 (秒) です。リクエストは、指定時間内に処理されないと失敗します。
session-manager-working-path	Yes	/tmp	操作に必要なファイルがブローカーにより書き込まれるディレクトリへのパスを指定します。このディレクトリにはブローカーしかアクセスできません。

Paramet Name	必須	デフォルト値	説明
enable-authorization-server	Yes	true	ブローカーをクライアント API の OAuth 2.0 アクセストークンの生成に使用される認証サーバーにするかどうかを指定します。
enable-authorization	Yes	true	クライアント認可を有効または無効にします。クライアント認可を有効にした場合、API リクエストの実行時にクライアント API からアクセストークンが提供される必要があります。クライアント認可を無効にすると、クライアント API によりアクセストークンなしでリクエストが実行されます。
enable-agent-authorizatic	Yes	true	エージェント認可を有効または無効にします。エージェント認可を有効にした場合、ブローカーとの通信時にエージェントからアクセストークンが提供される必要があります。

Parameter Name	必須	デフォルト値	説明
delete-session-duration-hours	No	1	削除されたセッションが非表示になって DescribeSession API コールによって返されなくなるまでの時間数を指定します。
connect-session-token-duration-minutes	No	60	ConnectSession トークンの有効状態が維持される時間を分単位で指定します。
client-to-broker-connect-https-port	Yes	8443	ブローカーによりクライアント接続に対してリッスンされる HTTPS ポートを指定します。
client-to-broker-connect-bind-host	No	0.0.0.0	ブローカーによりクライアント接続に対してバインドされるホストの IP アドレスを指定します。

Parameter Name	必須	デフォルト値	説明
client-to-broker-connector-key-store-file	Yes		TLS クライアント接続に使用されるキーストアを指定します。
client-to-broker-connector-key-store-pass	Yes		キーストアのパスを指定します。
agent-to-broker-connector-https-port	Yes	8445	ブローカーによりエージェント接続に対してリッスンされる HTTPS ポートを指定します。

Parameter Name	必須	デフォルト値	説明
agent-to-broker-connector-bind-host	No	0.0.0.0	ブローカーによりエージェント接続に対してバインドされるホストの IP アドレスを指定します。
agent-to-broker-connector-key-store-file	Yes		TLS エージェント接続に使用するキーストアを指定します。
agent-to-broker-connector-key-store-pass	Yes		キーストアのパスを指定します。
broker-to-broker-port	Yes	47100	ブローカー間接続に使用するポートを指定します。

Parameter Name	必須	デフォルト値	説明
broker-to-broker-bind-host	No	0.0.0.0	ブローカーによりブローカー間接続に対してバインドされるホストの IP アドレスを指定します。
broker-to-broker-discovery-port	Yes	47500	ブローカーの相互検出に使用するポートを指定します。

Parameter Name	必須	デフォルト値	説明
broker-to-broker-discovery-address	No		フリート内の他のブローカーの IP アドレスとポートを *ip_address* :*port* 形式で指定します。ブローカーが複数ある場合は、値をカンマで区切ります。broker-to-broker-discovery-multicast-group、broker-to-broker-discovery-multicast-port、broker-to-broker-discovery-AWS-region 、 または broker-to-broker-discovery-AWS-alb-target-group-arn を指定した場合は、このパラメータを省略します。

Parameter Name	必須	デフォルト値	説明
broker-to-broker-discovery-multicast-group	No		ブローカー間検出用のマルチキャストグループを指定します。broker-to-broker-discovery-addresses 、broker-to-broker-discovery-aws-region 、または broker-to-broker-discovery-AWS-alb-target-group-arn を指定した場合は、このパラメータを省略します。
broker-to-broker-discovery-multicast-port	No		ブローカー間検出用のマルチキャストポートを指定します。broker-to-broker-discovery-addresses 、broker-to-broker-discovery-AWS-region 、または broker-to-broker-discovery-AWS-alb-target-group-arn を指定した場合は、このパラメータを省略します。

Parameter Name	必須	デフォルト値	説明
broker-to-broker-discovery-AWS-region	No		ブローカー間検出に使用する Application Load Balancer の AWS リージョンを指定します。broker-to-broker-discovery-multicast-group 、broker-to-broker-discovery-multicast-port 、または broker-to-broker-discovery-addresses を指定した場合は、このパラメータを省略します。
broker-to-broker-discovery-AWS-alb-target-group-arn	No		ブローカー間検出に使用する Application Load Balancer ターゲットグループの ARN です。broker-to-broker-discovery-multicast-group、broker-to-broker-discovery-multicast-port 、または broker-to-broker-discovery-addresses を指定した場合は、このパラメータを省略します。

Parameter Name	必須	デフォルト値	説明
broker-to-broker-distributed-memory-max-size-mb	No	4096	NICE DCV セッションデータを保存するために単一ブローカーによって使用されるオフヒープメモリの最大量を指定します。
broker-to-broker-key-store-file	Yes		TLS ブローカー接続に使用するキーストアを指定します。
broker-to-broker-key-store-pass	Yes		キーストアのパスを指定します。
enable-cloud-watch-metrics	No	false	Amazon CloudWatch メトリクスを有効または無効にします。CloudWatch メトリクスを有効にした場合は、cloud-watch-region に対して値を指定する必要があります。

Parameter Name	必須	デフォルト値	説明
cloud-watch-region	No	enable-cloud-watch-metrics が true に設定されている場合のみ必須です。ブローカーが Amazon EC2 インスタンスにインストールされている場合、リージョンは IMDS から取得されます。	CloudWatch メトリクスが投稿される AWS リージョンです。
max-api-requests-per-second	No	1000	ブローカー API がスロットリングされる前に 1 秒間で処理できるリクエストの最大数を指定します。
enable-throttling-forwarded-for-header	No	false	true に設定されている場合、スロットリングにより X-Forwared-For ヘッダー (存在する場合) から発信者 IP が検索されます。
create-sessions-number-of-retries-on-failure	No	2	NICE DCV サーバーホストでセッション作成リクエストが失敗した場合、その後に実行される再試行回数の上限を指定します。失敗時に再試行されないようにする場合は 0 に設定します。

Paramet Name	必須	デフォルト値	説明
autorur f ile- argum ents- max- size	No	50	自動実行ファイルに渡される引数の最大数を指定します。
autorur f ile- argum ents- max- argumer length	No	150	各自動実行ファイルの最大長を文字数で指定します。
enable- pe rsister	Yes	false	true に設定されている場合、ブローカーのステータスデータは外部データベースに残されます。
persist ce- db	No	enable-persistence が true に設定されている場合のみ必須です。	残存のために使用するデータベースを指定します。サポートされる値は dynamodb と mysql のみです。
dynamo region	No	enable-persistence が true に設定されて persistence-db が dynamodb に設定されている場合のみ必須です。	DynamoDB テーブルの作成とアクセスが実行されるリージョンを指定します。

Paramet Name	必須	デフォルト値	説明
dynamoc table-rcu	No	enable-persistence が true に設定されて persistence-db が dynamodb に設定されている場合のみ必須です。	各 DynamoDB テーブルの読み取りキャパシティーユニット (RCU) を指定します。RCU の詳細については、「プロビジョンドキャパシティーの料金」を参照してください。
dynamoc table-wcu	No	enable-persistence が true に設定されて persistence-db が dynamodb に設定されている場合のみ必須です。	各 DynamoDB テーブルの書き込みキャパシティーユニット (WCU) を指定します。WCU の詳しい情報については、「プロビジョンドキャパシティーの料金」を参照してください。
dynamoc table-name-prefix	No	enable-persistence が true に設定されて persistence-db が dynamodb に設定されている場合のみ必須です。	各 DynamoDB テーブルに追加されるプレフィックスを指定します (同じ AWS アカウントを使用して複数のブローカークラスターを識別する際に便利です)。英数字、ドット、ダッシュ、下線のみを使用できます。

Parameter Name	必須	デフォルト値	説明
jdbc-connection-url	No	enable-persistence が true に設定されて persistence-db が mysql に設定されている場合のみ必須です。	MariaDB/MySQL データベースへの接続 URL を指定します。このデータベースには、エンドポイントとデータベース名が含まれます。URL は次の形式とします: `jdbc:mysql://<db_endpoint>:<db_port>/<db_name>?createDatabaseIfNotExist=true` <db_endpoint> が MariaDB/MySQL データベースエンドポイントである場合、<db_port> はデータベースポートになり、<db_name> はデータベース名になります。
jdbc-user	No	enable-persistence が true に設定されて persistence-db が mysql に設定されている場合のみ必須です。	MariaDB/MySQL データベースにアクセスできるユーザーの名前を指定します。
jdbc-password	No	enable-persistence が true に設定されて persistence-db が mysql に設定されている場合のみ必須です。	MariaDB/MySQL データベースにアクセスできるユーザーのパスワードを指定します。

Paramet Name	必須	デフォルト値	説明
seconds b efore- del eting- unr eachabl dcv- serve r	No	1800	アクセスできないサーバーがシステムから削除されるまでの秒数を指定します。

エージェント設定ファイル

エージェント設定ファイル (Linux の場合は /etc/dcv-session-manager-agent/agent.conf、Windows の場合は C:\Program Files\NICE\DCVSessionManagerAgent\conf\agent.conf) には、セッションマネージャー機能をカスタマイズするために設定できるパラメータが含まれています。希望するテキストエディタを使用して設定ファイルを手動で編集することができます。

下表にエージェント設定ファイルのパラメータを示します。

Paramet Name	必須	デフォルト値	説明
agent.b ker_hos	Yes		ブローカーホストの DNS 名を指定します。
agent.b ker_por	Yes	8445	ブローカーとの通信用のポートを指定します。
agent.c file	No		tls_strict が true に設定されている場合のみ必要です。TLS 証明書の検証に必要な証明書 (.pem)

Parameter Name	必須	デフォルト値	説明
			ファイルへのパスを指定します。ブローカーからエージェントに自己署名証明書をコピーしてください。
agent.init_folder	No	• /var/lib/dcv-session-manager-agent/init (Linux)	NICE DCV サーバーセッションを作成時に初期化できるカスタムスクリプトを保存するために使用するフォルダー (ホストサーバーにある) へのパスを指定します。絶対パスを指定する必要があります。このフォルダはアクセス可能な状態でなければならず、ファイルは、CreateSessions API の initFile リクエストパラメータを使用するユーザーが実行できる状態でなければなりません。
agent.tls_strict	No	true	厳密な TLS 検証を使用するかどうかを示します。
agent.software_statement_path	No		デフォルトのソフトウェアステートメントが使用されていない場合にのみ必要です。ソフトウェアステートメントファイルへのパスを指定します。詳細については、「generate-software-statement」を参照してください。

Paramet Name	必須	デフォルト値	説明
agent.t s_folde	No	• /etc/dcv-session-manager-agent (Linux) • C:\Program Files \NICE\DCVSess ionManagerAgent \conf\tags (Windows)	タグファイルが入っているフォルダへのパスを指定します。詳細については、「NICE DCV サーバーをターゲットにするためのタグの使用」を参照してください。
agent.a orun_fo er	No	• /var/lib/dcv-session-manage r-agent/autorun (Linux) • C:\ProgramData \NICE\DcvSess ionManagerAgent \autorun (Windows)	セッションのスタートアップ時に自動的に実行できるスクリプトとアプリを保存する場合に使用するフォルダー (ホストサーバーにある) へのパスを指定します。絶対パスを指定する必要があります。このフォルダはアクセス可能な状態でなければならず、ファイルは、CreateSessions API の AutorunFile リクエストパラメータを使用するユーザーが実行できる状態でなければなりません。
agent.m _virtua sessior	No	-1 (制限なし)	NICE DCV セッションマネージャーを使用して NICE DCV サーバーで作成できる仮想セッションの最大数です。

Parameter Name	必須	デフォルト値	説明
agent.max_concurrent_sessions_per_user	No	1	1 名のユーザーが NICE DCV セッションマネージャーを使用して NICE DCV サーバーで作成できる仮想セッションの最大数です。
agent.broker_update_interval	No	30	更新されたデータをブローカーに送信するまでに待機する秒数を指定します。送信データには、NICE DCV サーバーとホストのステータス、および更新されたセッション情報が含まれます。値が小さいほど、セッションマネージャーはエージェントが実行されているシステムで発生した変更をより認識できますが、システム負荷とネットワークトラフィックが増加します。値を高くするとシステムやネットワークの負荷は減りますが、セッションマネージャーはシステムの変更に反応しにくくなるため、120 よりも高い値は推奨されません。

Parameter Name	必須	デフォルト値	説明
log.level	No	info	ログファイルの詳細レベルを指定します。以下の詳細レベルを使用できます。 • error — 最小限の詳細を提供します。エラーのみが含まれています。 • warning — エラーと警告が含まれています。 • info — デフォルトの詳細レベルです。エラー、警告、情報メッセージが含まれています。 • debug — 最も詳細な情報を提供します。問題のデバッグに役立つ詳細情報を提供します。
log.directory	No	• /var/log/dcv-session-manager-agent/ (Linux) • C:\ProgramData\NICE\DCVSessionManagerAgent\log (Windows)	ログファイルを作成するディレクトリを指定します。

Parameter Name	必須	デフォルト値	説明
log.rotation	No	daily	ログファイルのローテーションを指定します。有効な 値は次のとおりです。 • hourly — ログファイルが 1 時間単位でローテーションされます。 • daily — ログファイルが 1 日単位でローテーションされます。
log.maxfile-size	No	10485760	ログファイルのサイズが指定されたサイズ (バイト) に達すると、ローテーションされます。新しいログファイルが作成され、そのファイルにさらにログイベントが記録されます。
log.rotate	No	9	ローテーションで保持されるログファイルの最大数ローテーションが発生してこの数に達するたびに、最も古いログファイルが削除されます。

NICE DCV セッションマネージャーのリリースノートとドキュメント履歴

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

トピック

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

NICE DCV セッションマネージャーのリリースノート

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

トピック

- 2023.1-16388 — 2024 年 6 月 26 日
- 2023.1— 2023 年 11 月 9 日
- 2023.0-15065— 2023 年 5 月 4 日
- 2023.0-14852— 2023 年 3 月 28 日
- 2022.2-13907— 2022 年 11 月 11 日
- 2022.1-13067— 2022 年 6 月 29 日
- 2022.0-11952 — 2022 年 2 月 23 日
- 2021.3-11591 — 2021 年 12 月 20 日
- 2021.2-11445 — 2021 年 11 月 18 日
- 2021.2-11190 — 2021 年 10 月 11 日
- 2021.2-11042 — 2021 年 9 月 1 日
- 2021.1-10557 — 2021 年 5 月 31 日
- 2021.0-10242 — 2021 年 4 月 12 日
- 2020.2-9662 — 2020 年 12 月 4 日

- 2020.2-9508 — 2020 年 11 月 11 日

2023.1-16388 — 2024 年 6 月 26 日

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

2023.1— 2023 年 11 月 9 日

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

2023.0-15065— 2023 年 5 月 4 日

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

2023.0-14852— 2023 年 3 月 28 日

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

ビルド番号	変更とバグ修正
• CLI: 132	

2022.2-13907— 2022 年 11 月 11 日

ビルド番号	変更とバグ修正
• ブローカー: 382 • エージェント: 612 • CLI: 123	• `DescribeSessions` の応答に `Substate` フィールドが追加されました。 • 使用中の URL に応じて CLI がブローカーに接続できない問題を修正しました。

2022.1-13067— 2022 年 6 月 29 日

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

2022.0-11952 — 2022 年 2 月 23 日

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

2021.3-11591 — 2021 年 12 月 20 日

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

2021.2-11445 — 2021 年 11 月 18 日

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

2021.2-11190 — 2021 年 10 月 11 日

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

2021.2-11042 — 2021 年 9 月 1 日

ビルド番号	新機能	変更とバグ修正
• ブローカー: 254	• NICE DCV セッションマネージャーで、コマンドラインインターフェイス (CLI) がサポートされるようになりました。API を呼び出すのではな	• 外部認可サーバーを登録するときに、認可サーバーで JSON 形式のウェブトークンの署名に使用されるアルゴリズムを指定できるようにな

ビルド番号	新機能	変更とバグ修正
• エージェント: 413 • CLI: 37	く、CLI で、NICE DCV セッションの作成と管理を実行できます。 • NICE DCV セッションマネージャーにブローカーデータの永続性を導入しました。可用性を高めるために、ブローカーでは、サーバーの状態情報を外部データストアに残しておいて、スタートアップ時にデータを復元することができます。	りました。この変更により、Azure AD を外部認可サーバーとして使用できます。

2021.1-10557 — 2021 年 5 月 31 日

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

2021.0-10242 — 2021 年 4 月 12 日

ビルド番号	変更とバグ修正

• ブローカー: 183
• エージェント: 318
	• NICE DCV セッションマネージャーに次の新しい API を導入しました。
	  • OpenServers
	  • CloseServers
	  • DescribeServers
	  • GetSessionScreenshots

ビルド番号	変更とバグ修正

	• 次の新しい設定パラメータも導入しました。
	  • ブローカーパラメータ: session-screenshot-max-width 、session-screenshot-max-height 、session-screenshot-format 、create-sessions-queue-max-size 、create-sessions-queue-max-time-seconds
	  • エージェントパラメータ: agent.autorun_folder 、max_virtual_sessions 、max_concurrent_sessions_per_user
	
	  エージェントパラメータ: agent.autorun_folder 、max_virtual_sessions 、max_concurrent_sessions_per_user
	
	  エージェントパラメータ: agent.autorun_folder 、max_virtual_sessions 、max_concurrent_sessions_per_user

2020.2-9662 — 2020 年 12 月 4 日

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

2020.2-9508 — 2020 年 11 月 11 日

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

ドキュメント履歴

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

変更	説明	日付
NICE DCV バージョン 2023.1-16388	NICE DCV セッションマネージャが NICE DCV 2023.1-16388 用に更新されました。詳細については、「2023.1-16388 — 2024 年 6 月 26 日」を参照してください。	2024 年 6 月 26 日
NICE DCV バージョン 2023.1	NICE DCV 2023.1 用に NICE DCV セッションマネージャーが更新されました。詳細については、「2023.1— 2023 年 11 月 9 日」を参照してください。	2023 年 11 月 9 日
NICE DCV バージョン 2023.0	NICE DCV 2023.0 用に NICE DCV セッションマネージャーが更新されました。詳細については、「2023.0-14852 — 2023 年 3 月 28 日」を参照してください。	2023 年 3 月 28 日
NICE DCV バージョン 2022.2	NICE DCV 2022.2 用に NICE DCV セッションマネージャーが更新されました。詳細については、「2022.2-13907 — 2022 年 11 月 11 日」を参照してください。	2022 年 11 月 11 日
NICE DCV バージョン 2022.1	NICE DCV 2022.1 用に NICE DCV セッションマネージャーが更新されました。詳細については、「2022.1-13067 — 2022 年 6 月 29 日」を参照してください。	2022 年 1 月 29 日
NICE DCV バージョン 2022.0	NICE DCV 2022.0 用に NICE DCV セッションマネージャーが更新されました。詳細については、「2022.0-11952	2022 年 2 月 23 日

変更	説明	日付
	— 2022 年 2 月 23 日」を参照してください。	
NICE DCV バージョン 2021.3	NICE DCV 2021.3 用に NICE DCV セッションマネージャーが更新されました。詳細については、「2021.3-11591 — 2021 年 12 月 20 日」を参照してください。	2021 年 12 月 20 日
NICE DCV バージョン 2021.2	NICE DCV 2021.2 用に NICE DCV セッションマネージャーが更新されました。詳細については、「2021.2-11042 — 2021 年 9 月 1 日」を参照してください。	2021 年 9 月 1 日
NICE DCV バージョン 2021.1	NICE DCV 2021.1 用に NICE DCV セッションマネージャーが更新されました。詳細については、「2021.1-10557 — 2021 年 5 月 31 日」を参照してください。	2021 年 5 月 31 日
NICE DCV バージョン 2021.0	NICE DCV 2021.0 用に NICE DCV セッションマネージャーが更新されました。詳細については、「2021.0-10242 — 2021 年 4 月 12 日」を参照してください。	2021 年 4 月 12 日
NICE DCV セッションマネージャーの初回リリース。	このコンテンツの初版です。	2020 年 11 月 11 日

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