aws

管理者ガイド

Amazon DCV Session Manager

Amazon DCV Session Manager: 管理者ガイド

Table of Contents

Amazon DCV Session Manager とは

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

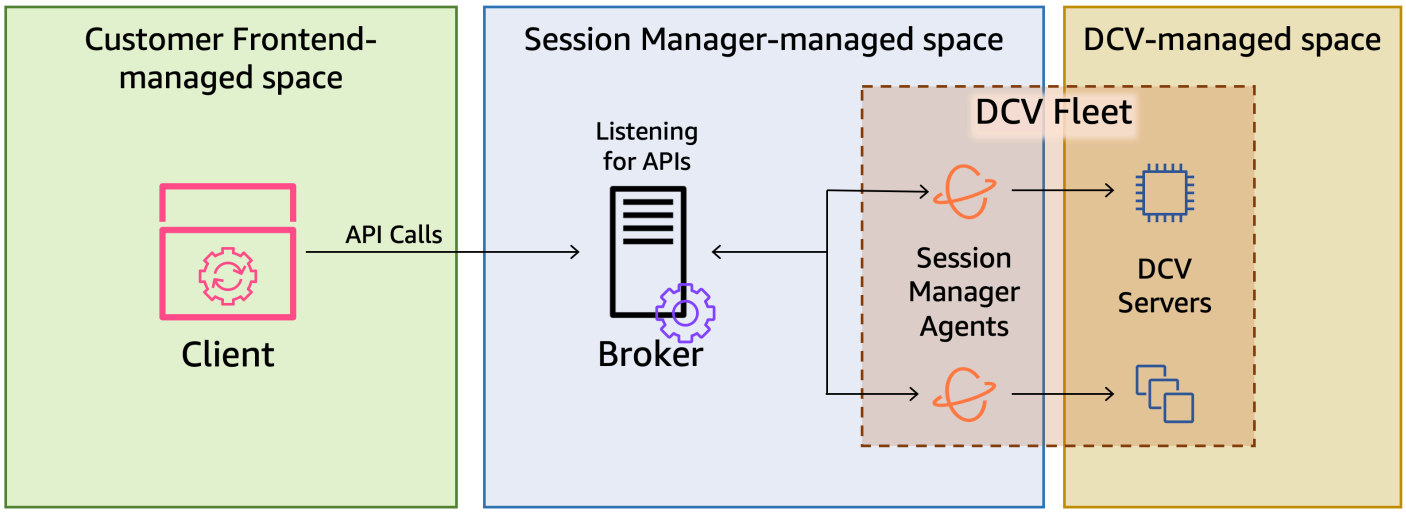
Amazon DCV Session Manager は、インストール可能なソフトウェアパッケージ (エージェントとブローカー) とアプリケーションプログラミングインターフェイス (API) のセットで、デベロッパーや独立系ソフトウェアベンダー (ISVs) が、Amazon DCVサーバーのフリート全体で Amazon DCVセッションのライフサイクルをプログラムで作成および管理できるフロントエンドアプリケーションを簡単に構築できます。

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。Session Manager の使用の詳細についてはAPIs、「Amazon DCV Session Manager 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

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、Session Manager をホストして公開するウェブサーバーですAPIs。クライアントから Amazon DCVセッションを管理するAPIリクエストを受信して処理し、その指示を関連するエージェント に渡します。ブローカーは Amazon DCVサーバーとは別のホストにインストールする必要がありますが、クライアントがアクセスでき、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 Amazon DCVサーバーにインストールされます。エージェントはブローカーから指示を受け取り、それぞれの Amazon DCVサーバーで実行します。エージェントは Amazon DCVサーバーの状態もモニタリングし、ステータスの更新を定期的にブローカーに送信します。

APIs

Session Manager は、Amazon DCVサーバーのフリートで Amazon DCVセッションを管理するために使用できる一連のRESTアプリケーションプログラミングインターフェイス (APIs) を公開します。APIs は ブローカー でホストされ、公開されます。デベロッパーは、 を呼び出すカスタムセッション管理クライアントを構築できますAPIs。

クライアント

クライアントは、ブローカー によって公開される Session Manager を呼び出すために開発APIsするフロントエンドアプリケーションまたはポータルです。エンドユーザーはクライアントを使用して、フリート内の Amazon DCVサーバーでホストされているセッションを管理します。

アクセストークン

API リクエストを行うには、アクセストークンを指定する必要があります。トークンは、登録済みクライアント によってブローカーまたは外部認証サーバーからリクエストできますAPIs。トークンをリクエストしてアクセスするには、クライアントが有効な認証情報を提供するAPI必要があります。

クライアント API

クライアントAPIは、Session Manager API 定義YAMLファイルから Swagger Codegen を使用して生成されます。クライアントAPIはAPIリクエストを行うために使用されます。

Amazon DCVセッション

クライアントが接続できる Amazon DCVサーバーに Amazon DCVセッションを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 Amazon DCVサーバーに接続できます。Amazon DCVは、コンソールセッションと仮想セッションをサポートしています。Session Manager を使用してAPIs、Amazon DCVセッションのライフサイクルを管理します。Amazon DCVセッションは、次のいずれかの状態になります。

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

機能

DCV Session Manager には以下の機能があります。

- Amazon DCVセッション情報 — 複数の Amazon DCVサーバーで実行されているセッションに関する情報を取得します。
- 複数の Amazon DCVセッションのライフサイクルを管理する — 1 つのAPIリクエストで複数の Amazon DCVサーバーにまたがる複数のユーザーの複数のセッションを作成または削除します。
- タグをサポート — セッションの作成時にカスタムタグを使用して Amazon DCVサーバーのグループをターゲットにします。

- 複数の Amazon DCVセッションのアクセス許可を管理します。1 つのAPIリクエストで複数のセッションのユーザーアクセス許可を変更します。
- 接続情報 — Amazon DCVセッションのクライアント接続情報を取得します。
- 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。Amazon EC2インスタンスDCVで Amazon を実行している場合は、Amazon EC2 Auto Scaling などの追加 AWS サービスを使用してインフラストラクチャのスケーリングを管理する必要がある場合があります。

料金

Session Manager は、EC2インスタンスを実行している AWS お客様に無料でご利用いただけます。

オンプレミスのお客様は、Amazon DCV Plus または Amazon DCV Professional Plus ライセンスが必要です。Amazon DCV Plus または Amazon DCV Professional Plus ライセンスの購入方法については、「Amazon DCVウェブサイトで購入する方法」および「お住まいのリージョンの Amazon DCV ディストリビューターまたはリセラーを検索する」を参照してください。すべてのオンプレミスのお客様に Amazon DCV Session Manager を試すことができるように、ライセンス要件は Amazon DCVバージョン 2021.0 以降でのみ適用されます。

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

Amazon DCV Session Manager の要件

Amazon DCV Session Manager 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 Stream 9
	• RHEL 7.6 以降
	• RHEL 8.x
		• Windows
		  • Windows Server 2022
		  • Windows Server 2019

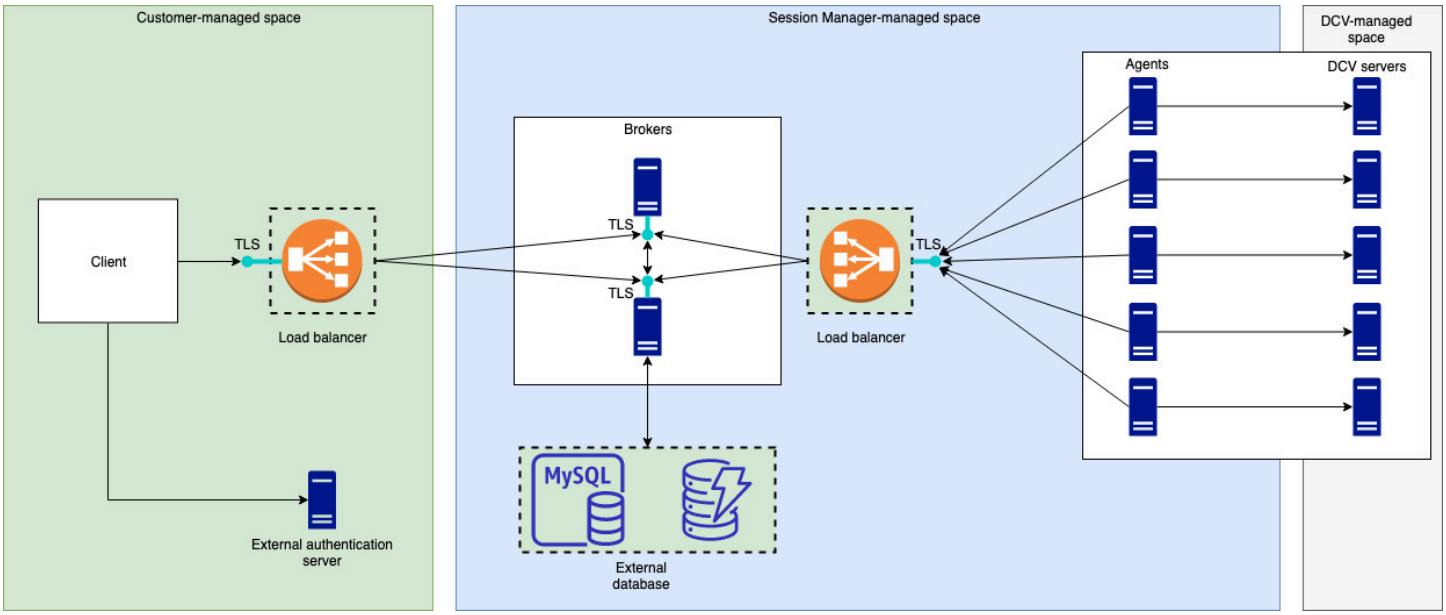
	ブローカー	エージェント

	• RHEL 9.x
	• Rocky Linux 8.5 以降
	• Rocky Linux 9.x
	• Ubuntu 20.04
	• Ubuntu 22.04
	• Ubuntu 24.04
		• Windows Server 2016
		• [Linux サーバー]
		  • Amazon Linux 2
		  • CentOS Stream 9
		  • RHEL 8.x
		  • RHEL 9.x
		  • Rocky Linux 8.5 以降
		  • Rocky Linux 9.x
		  • Ubuntu 20.04
		  • Ubuntu 22.04
		  • Ubuntu 24.04
		  • SUSE Linux Enterprise 12 SP4以降
		  • 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 9.x、8.x/9.x RHEL および Rocky 8.x/9.x のみ) • 64 ビット ARM (Ubuntu 22.04 および 24.04)
「メモリ」	8 GB	4 GB
Amazon DCVバージョン	Amazon DCV 2020.2 以降	Amazon 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、Amazon DCVサーバー上のエージェントとネットワーク接続している必要があります。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、Amazon 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、相互に通信できる必要があります。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 、および MySQL がサポートされています。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で永続的な双方向HTTPs接続を開始できる必要があります。

を呼び出すには、クライアント またはフロントエンドアプリケーションがブローカーにアクセスできる必要がありますAPIs。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。

Amazon DCV Session Manager 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 Session Manager 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。

Amazon DCV Session Manager を設定するには、以下を実行します。

ステップ

- ステップ 1: Amazon DCVサーバーを準備する
- ステップ 2: Amazon DCV Session Manager ブローカーを設定する
- ステップ 3: Amazon DCV Session Manager エージェントを設定する
- ステップ 4: ブローカーを認証DCVサーバーとして使用するように Amazon 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: Amazon DCVサーバーを準備する

Session Manager を使用する予定の Amazon DCVサーバーのフリートが必要です。Amazon DCV サーバーのインストールの詳細については、「Amazon DCV管理者ガイド」の「Amazon DCVサーバーのインストール」を参照してください。

Linux Amazon DCVサーバーでは、Session Manager は という名前のローカルサービスユーザーを使用しますdcvsmagent。このユーザーは、Session Manager 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。このサービスユーザーに Amazon の管理者権限を付与DCVして、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 Amazon DCVサーバーのローカルサービスユーザーを追加するには

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. Amazon DCVサーバーを停止して再起動します。

Session Manager は、Amazon DCVサーバーに既に存在するユーザーに代わってのみ Amazon DCV セッションを作成できます。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、対象となる各エンドユーザーが Amazon DCVサーバーに有効なシステムユーザーを持っ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 Amazon DCVサーバーを使用する場合は、次の手順を実行して、完了した設定でホストの Amazon マシンイメージ (AMI) を作成し、 を使用して残りのブローカーと Amazon サーバーを起動AMIsすることで、エージェントDCVで 1 つのブローカーと 1 つの Amazon DCVサーバーのみを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コマンドをリモートで実行することもできます。

ステップ 2: Amazon DCV Session Manager ブローカーを設定する

ブローカーは 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、「Amazon DCV Session Manager の要件」を参照してください。エージェントと Amazon 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必要があります。

ブローカーをインストールして起動するには

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署名でデジタルGPG署名されます。パッケージマネージャーがパッケージ署名を検証できるようにするには、Amazon DCVGPGキーをインポートする必要があります。次のコマンドを実行して Amazon DCVGPGキーをインポートします。

   - Amazon Linux 2、RHEL、CentOS 、および 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、RHEL7.x

   ```
   $ wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerBrokers/nice-dcv-session-manager-broker-2024.0.457-1.el7.noarch.rpm
   ```

   - RHEL 8.x、および Rocky Linux 8.x

   ```
   $ wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerBrokers/nice-dcv-session-manager-broker-2024.0.457-1.el8.noarch.rpm
   ```

   - CentOS 9.x、RHEL9.x、および Rocky Linux 9.x

   ```
   $ wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerBrokers/nice-dcv-session-manager-broker-2024.0.457-1.el9.noarch.rpm
   ```

   - Ubuntu 20.04

   ```
   $ wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerBrokers/nice-dcv-session-manager-broker_2024.0.457-1_all.ubuntu2004.deb
   ```

   - Ubuntu 22.04

   ```
   $ wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerBrokers/nice-dcv-session-manager-broker_2024.0.457-1_all.ubuntu2204.deb
   ```

   - Ubuntu 24.04

   ```
   $ wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerBrokers/nice-dcv-session-manager-broker_2024.0.457-1_all.ubuntu2404.deb
   ```

4. パッケージをインストールします。

   - Amazon Linux 2、RHEL7.x

   ```
   $ sudo yum install -y ./nice-dcv-session-manager-broker-2024.0.457-1.el7.noarch.rpm
   ```

- RHEL 8.x および Rocky Linux 8.x

```
$ sudo yum install -y ./nice-dcv-session-manager-
broker-2024.0.457-1.el8.noarch.rpm
```

- Ubuntu 20.04

```
$ sudo apt install -y ./nice-dcv-session-manager-
broker_2024.0.457-1_all.ubuntu2004.deb
```

- Ubuntu 22.04

```
$ sudo apt install -y ./nice-dcv-session-manager-
broker_2024.0.457-1_all.ubuntu2204.deb
```

- Ubuntu 24.04

```
$ sudo apt install -y ./nice-dcv-session-manager-
broker_2024.0.457-1_all.ubuntu2404.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
```

そうでない場合は、ブローカーが適切な Java バージョンをターゲットにするために使用する Java ホームディレクトリを明示的に設定できます。これにより、ブローカー設定ファイルの パラメータの設定が完了broker-java-homeします。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起動し、インスタンスが起動するたびに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配置します。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: Amazon DCV Session Manager エージェントを設定する

エージェントは、フリート内のすべての Amazon DCVサーバーホストにインストールする必要があります。エージェントは、Windows サーバーと Linux サーバーの両方に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、「Amazon DCV Session Manager 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、Amazon 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 エージェントは、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。
> 以下の手順は、64 ビット x86 ホストにエージェントをインストールするためのものです。エージェントを 64 ビットARMホストにインストールするには、 を置き換えます *x86_64* `aarch64` で。Ubuntu の場合は、*amd64* `arm64` で。

Linux ホストにエージェントをインストールするには

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

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

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent-2024.0.781-1.el7.x86_64.rpm
```

- RHEL 8.x および Rocky Linux 8.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent-2024.0.781-1.el8.x86_64.rpm
```

- CentOS 9.x、RHEL9.x、および Rocky Linux 9.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent-2024.0.781-1.el9.x86_64.rpm
```

- Ubuntu 20.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent_2024.0.781-1_amd64.ubuntu2004.deb
```

- Ubuntu 22.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent_2024.0.781-1_amd64.ubuntu2204.deb
```

- Ubuntu 24.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent_2024.0.781-1_amd64.ubuntu2404.deb
```

- SUSE Linux Enterprise 12

```
$ curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent-2024.0.781-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$ curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent-2024.0.781-1.sles15.x86_64.rpm
```

3. パッケージをインストールします。

   - Amazon Linux 2 および RHEL 7.x

   ```
   $  sudo yum install -y ./nice-dcv-session-manager-
   agent-2024.0.781-1.el7.x86_64.rpm
   ```

   - RHEL 8.x および Rocky Linux 8.x

   ```
   $  sudo yum install -y ./nice-dcv-session-manager-
   agent-2024.0.781-1.el8.x86_64.rpm
   ```

   - CentOS 9.x、RHEL9.x、および Rocky Linux 9.x

   ```
   $  sudo yum install -y ./nice-dcv-session-manager-
   agent-2024.0.781-1.el9.x86_64.rpm
   ```

   - Ubuntu 20.04

   ```
   $  sudo apt install ./nice-dcv-session-manager-
   agent_2024.0.781-1_amd64.ubuntu2004.deb
   ```

   - Ubuntu 22.04

   ```
   $  sudo apt install ./nice-dcv-session-manager-
   agent_2024.0.781-1_amd64.ubuntu2204.deb
   ```

   - Ubuntu 24.04

   ```
   $  sudo apt install ./nice-dcv-session-manager-
   agent_2024.0.781-1_amd64.ubuntu2404.deb
   ```

   - SUSE Linux Enterprise 12

   ```
   $ sudo zypper install ./nice-dcv-session-manager-
   agent-2024.0.781-1.sles12.x86_64.rpm
   ```

   - SUSE Linux Enterprise 15

```
$ sudo zypper install ./nice-dcv-session-manager-
agent-2024.0.781-1.sles15.x86_64.rpm
```

4. ブローカーの自己署名証明書 (前のステップでコピーした) のコピーをエージェントの /etc/dcv-session-manager-agent/ ディレクトリに配置します。
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。

   - ではbroker_host、ブローカーがインストールされているホストDNSの名前を指定します。

     > **Important**
     >
     > ブローカーが Amazon EC2インスタンスで実行broker_hostされている場合は、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) ではbroker_port、ブローカーと通信する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はポート を介して通信します8445。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、ブローカーが同じポートを使用するように設定されていることを確認します。
   - 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'
     ```

     または、TLS検証を無効にする場合は、 tls_strict を に設定しますfalse。
6. ファイルを保存して閉じます。
7. 次のコマンドを実行してエージェントを起動します。

   ```
   $ 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には

1. エージェントインストーラ 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 画面でライセンス契約を注意深く読み、同意する場合は、条件に同意し、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 (前のステップでコピーした) のコピーをエージェントの `C:\Program Files\NICE\DCVSessionManagerAgent\conf\`フォルダに配置します。
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。

   - では`broker_host`、ブローカーがインストールされているホストDNSの名前を指定します。

     > **Important**
     >
     > ブローカーが Amazon EC2インスタンスで実行`broker_host`されている場合は、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) では`broker_port`、ブローカーと通信する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はポート を介して通信します`8445`。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、ブローカーが同じポートを使用するように設定されていることを確認します。
   - `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'
     ```

     または、TLS検証を無効にする場合は、 `tls_strict` を に設定します`false`。
7. ファイルを保存して閉じます。
8. 変更を有効にするには、エージェントサービスを停止して再起動します。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
   ```

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

ステップ 4: ブローカーを認証DCVサーバーとして使用するように Amazon サーバーを設定する

ブローカーDCVをクライアント接続トークンを検証するための外部認証サーバーとして使用するように Amazon 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 Amazon DCVサーバーを設定する必要があります。

Linux Amazon DCV server

Linux Amazon DCVサーバーのローカルサービスユーザーを追加するには

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。

   - ではca-file、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。
   - ではauth-token-verifier、ブローカーのURLトークン検証子に を次の形式で指定します。 https://*broker_ip_or_dns*:*port*/agent/validate-authentication-tokenブローカーエージェント通信に使用され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 IP アドレス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. Amazon DCVサーバーを停止して再起動します。詳細については、「Amazon 管理者ガイド」の「Amazon DCV サーバーの停止」と「Amazon DCVサーバーの起動」を参照してください。 DCV

Windows Amazon DCV server

Windows Amazon 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。
3. 値データ には、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。

4. auth-token-verifier パラメータを開きます。
5. 値データ の場合、ブローカーのトークン検証子URLの を次の形式で指定します。`https://broker_ip_or_dns:port/agent/validate-authentication-token`
6. ブローカーエージェント通信に使用され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 IP アドレス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。

7. [OK] を選択して Windows レジストリエディタを閉じます。
8. Amazon DCVサーバーを停止して再起動します。詳細については、「Amazon 管理者ガイド」の「Amazon DCV サーバーの停止」と「Amazon DCVサーバーの起動」を参照してください。 DCV

ステップ 5: インストールを検証する

エージェントをセットアップし、ブローカーをセットアップし、Amazon DCVサーバーで両方を設定したら、インストールが正しく機能していることを確認する必要があります。

トピック

エージェントを検証する

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り、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されていて、ブローカーに接続できる場合、 コマンドは を返すはずです`success`。

コマンドで異なる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確認してください。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

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

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されており、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。

ブローカーを検証する

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り、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可能であることを確認してください。

ブローカーにアクセス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、ブローカーは以下を返します。

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

Amazon DCV Session Manager の設定

シームレスで安全なエクスペリエンスを提供するには、組織のニーズと要件に応じて Session Manager を適切に設定することが重要です。このセクションでは、ユーザーアクセスの管理、ネットワーク設定の設定、セッション設定のカスタマイズなど、Session Manager のセットアップと設定に関連する主要なステップ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

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、1 つのエージェントとブローカーホストのみをインストールして設定し、それらのホストから Amazon マシンイメージ (AMI) を作成し、残りのホストを から起動することをお勧めしますAMIs。

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 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 を使用するアクセス許可を付与するインスタンスプロファイルをブローカーホストとエージェントホストにアタッチする必要がありますAPIs。詳細については、IAM「Amazon ユーザーガイド」の「Amazon のロールEC2」を参照してください。 EC2

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、「 ユーザーガイド」のIAM「ロールの作成」を参照してください。 IAM

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create-instance-profile」の「」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「インスタンスへのIAMロールのアタッチ」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

ロードバランサーリスナーHTTPSに を使用する場合は、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) を作成しますPCA。詳細については、 AWS 「 Certificate Manager プライベート認証機関ユーザーガイド」の「CA の作成手順」を参照してください。
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、 AWS 「 Certificate Manager プライベート認証機関ユーザーガイド」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。

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

   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 プロトコル で を選択しHTTPS、Load Balancer ポート で と入力します8443。
   d. ではVPC、VPC使用する を選択し、その 内のすべてのサブネットを選択しますVPC。
   e. [Next (次へ)] を選択します。
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. 証明書タイプ では、 から証明書を選択する ACM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. [Next (次へ)] を選択します。
4. ステップ 3: セキュリティグループ を設定するか、新しいセキュリティグループを作成するか、フロントエンドクライアントと HTTPSおよび ポート 8443 経由の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を許可する既存のセキュリティグループを選択します。

   [Next (次へ)] を選択します。
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. [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。プロトコルバージョン では、 を選択しますHTTP1。
e. ヘルスチェックプロトコル で を選択しHTTPS、パス で を入力します/health。
f. [Next (次へ)] を選択します。

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

最初の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、ブローカーAMIから を作成し、 を使用して残りのブローカーAMI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: Amazon DCV Session Manager ブローカーを設定する」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. にbroker-to-broker-discovery-aws-alb-target-group-arn、ブローカーロードバランサーに関連付けられたARNターゲットグループの 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AMIから を作成します。詳細については、「Linux インスタンス用 Amazon ユーザーガイド」の「インスタンスAMIから Linux を作成する」を参照してください。 EC2
5. を使用してAMI、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. で Amazon EC2コンソールを開きますhttps://console.aws.amazon.com/ec2/。

   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 プロトコル で を選択しHTTPS、Load Balancer ポート で と入力します8445。
   d. ではVPC、VPC使用する を選択し、その 内のすべてのサブネットを選択しますVPC。
   e. [Next (次へ)] を選択します。
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. 証明書タイプ で、 から証明書を選択する ACM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. [Next (次へ)] を選択します。

4. ステップ 3: セキュリティグループ を設定するか、新しいセキュリティグループを作成するか、エージェントとブローカーが HTTPSおよび ポート 8445 経由で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を許可する既存のセキュリティグループを選択します。

   [Next (次へ)] を選択します。

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. [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。プロトコルバージョン では、 を選択しますHTTP1。

   e. ヘルスチェックプロトコル で を選択しHTTPS、パス で を入力します/health。

   f. [Next (次へ)] を選択します。

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、エージェントAMIから を作成し、 を使用して残りのエージェントAMI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. Amazon DCVサーバーを準備します。詳細については、「ステップ 1: Amazon 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: Amazon DCV Session Manager エージェントを設定する」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、エージェントのロードバランサーの DNS
> - `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DCVサーバーとして使用するように Amazon サーバーを設定します。詳細については、「ステップ 4: ブローカーを認証DCVサーバーとして使用するように Amazon サーバーを設定する」を参照してください。

> **Important**
>
> Amazon DCVサーバー設定ファイルを変更する場合：
>
> - `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、 エージェントロードバランサーの を使用しますDNS。*`broker_ip_or_dns`*

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AMIから を作成します。詳細については、「Linux インスタンス用 Amazon ユーザーガイド」の「インスタンスAMIから Linux を作成する」を参照してください。 EC2
7. を使用して残りの エージェントAMIを起動し、作成したインスタンスプロファイルを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。

タグを使用して Amazon DCVサーバーをターゲットにする

Session Manager エージェントにカスタムタグを割り当てると、それらとそれらが関連付けられている Amazon DCVサーバーを識別して分類できます。新しい Amazon DCVセッションを作成するときは、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 Amazon DCVサーバーのグループをターゲットにできます。エージェントタグに基づいて Amazon DCVサーバーをターゲットにする方法の詳細については、「 Session Manager デベロッパーガイドCreateSessionRequests 」の「」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 で `.toml`、ファイルの内容は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

認証サーバーは、クライアントとエージェントの認証SDKsと承認を担当するサーバーです。

デフォルトでは、Session Manager はブローカーを認証サーバーとして使用し、エージェント用のクライアントSDKsおよびソフトウェアステートメント用の OAuth 2.0 アクセストークンを生成します。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 コマンドを使用して、プール名とその作成先の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userPoolIdのドメイン名と 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userPoolIdの 、クライアントの名前、および作成先の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するときは、この情報をデベロッパーに提供する必要があります。

4. ユーザープール用に新しい OAuth2.0 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userPoolIdの 、リソースサーバーの一意の識別子と名前、スコープ、および作成先の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。Cognito ユーザープールを操作するときにクライアントがOAuthプロトコルに従うことが許可されていることを示す--allowed-o-auth-flows-user-pool-clientには、 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、URL認証サーバーの は ですhttps://cognito-idp.*us-east-1*.amazonaws.com/*us-east-1_QLEXAMPLE*/.well-known/jwks.json。

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URL生成されたクライアント ID、クライアントシークレット、およびサーバーを指定します。アクセストークンのリクエストの詳細については、「Amazon DCV Session Manager デベロッパーガイド」の「アクセストークンの取得とAPIリクエストの作成」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、キーを再生成したり、Amazon DCV Server が開いているか閉じられているかに関する情報を失うことなく、クラスターを停止して再起動することもできます。

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
- Amazon DCVサーバーのステータス

Amazon DCV Session Manager は、DynamoDB 、MariaDB 、および MySQL データベースをサポートしています。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 でホストされている場合はEC2、追加のセットアップを必要としないため、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。
   - では、各テーブルがサポートする読み取りキャパシティーユニット (RCU) の量`dynamodb-table-rcu`を指定します。の詳細についてはRCU、DynamoDB プロビジョンドキャパシティ」を参照してください。
   - では、各テーブルがサポートする書き込みキャパシティーユニット (WCU) の量`dynamodb-table-wcu`を指定します。の詳細についてはWCU、DynamoDB プロビジョニングされた容量」を参照してください。
   - では、各 dynamodb-table-name-prefixDynamoDB テーブルに追加す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区別するのに役立ちます) 。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
```

ブローカーホストには、DynamoDB を呼び出すアクセス許可が必要ですAPIs。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 /My に保持するようにブローカーを設定するSQL

> **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 /My でデータの保持を開始するようにブローカーを設定しますSQL。

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
```

Amazon DCV Connection Gateway との統合

Amazon DCV Connection Gateway は、ユーザーが LANまたは への単一のアクセスポイントを介して Amazon DCVサーバーの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VPC。

インフラストラクチャに Amazon DCV Connection Gateway 経由でアクセスできる Amazon DCV サーバーが含まれている場合は、Amazon DCV Connection Gateway を統合するように Session Manager を設定できます。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。Connection Gateway はエンドポイントをAPI呼び出し、ブローカーによって選択されたホストに Amazon 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。

トピック

- Amazon DCV Connection Gateway のセッションリゾルバーとして Session Manager ブローカーを設定する
- オプション - TLSクライアント認証を有効にする
- Amazon DCV Session Manager Amazon DCVサーバー - DNSマッピングリファレンス

Amazon DCV Connection Gateway のセッションリゾルバーとして Session Manager ブローカーを設定する

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
- 無料TCPポート`gateway-to-broker-connector-https-port`に設定する (デフォルトは 8447)
- Amazon DCV Connection Gateway 接続のためにブローカーがバインドするホストの IP アドレス`gateway-to-broker-connector-bind-host`に設定します (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
```

次のステップで Amazon DCV Connection Gateway をインストールするときに必要になります。

Amazon DCV Connection Gateway 側

- Amazon DCV Connection Gateway ドキュメントの セクションに従ってください。

  Amazon DCV Connection Gateway はブローカーをHTTPAPI呼び出すため、ブローカーが自己署名証明書を使用している場合は、ブローカー証明書を Amazon DCV Connection Gateway ホスト (前のステップで取得) にコピーし、Amazon DCV Connection Gateway 設定の`[resolver]`セクションで `ca-file`パラメータを設定する必要があります。

オプション - TLSクライアント認証を有効にする

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。セキュアチャネルを確立する前に Session Manager ブローカーが Connection Gateway の ID も検証する必要がある場合は、次のセクションの手順に従ってクライアントTLS認証機能を有効にする必要があります。

> **Note**
>
> Session Manager がロードバランサーの背後にある場合、Application Load Balancer (ALBs) や Gateway Load Balancer () など、TLS接続が終了しているロードバランサーではTLSクライアント認証を有効にすることはできませんGLBs。Network Load Balancer () など、TLS終了のないロードバランサーのみをサポートできますNLBs。ALBs または を使用する場合 GLBs、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連絡できるように強制して、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、こちら: の セキュリティグループ VPC

セッションマネージャーブローカー側

1. Session Manager ブローカーと Amazon DCV Connection Gateway 間の通信でTLSクライアント認証を有効にするには、次のステップに従ってください。
2. 実行して必要なキーと証明書を生成する: コマンドの出力は、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を通知します。

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

3. Amazon DCV Connection Gateway 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配置します。次のステップで Amazon DCV Connection Gateway 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。
   - 前のステップで作成した TrustStore ファイルのパス`gateway-to-broker-connector-trust-store-file`に設定する
   - 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`gateway-to-broker-connector-trust-store-pass`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
```

Amazon DCV Connection Gateway 側

- Amazon DCV Connection Gateway 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

Amazon DCV Session Manager Amazon DCVサーバー - DNSマッピングリファレンス

Amazon DCV Connection Gateway は、DCVサーバーインスタンスに接続するために Amazon DCV サーバーDNSの名前を必要とします。このセクションでは、各DCVサーバーと関連付けられたDNS名前間のマッピングを含むJSONファイルを定義する方法について説明します。

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"
  }
 },
 ...
]
```

コードの説明は以下のとおりです。

`ServerIdType:`

値が参照する ID のタイプを識別します。現在、使用可能な値は ipAddress、 agentServerId、および ですinstanceId。

`Ip:`

Amazon インフラストラクチャEC2とオンプレミスインフラストラクチャの両方で使用できます。ifconfig (Linux) または ipconfig (Windows) コマンドを使用して、システム管理者がすばやく取得できます。この情報は、レスポンスでも利用できます DescribeServers API。

`Id:`

Amazon インフラストラクチャEC2とオンプレミスインフラストラクチャの両方で使用できます。Session Manager エージェントは、ホスト名または IP アドレスが変更されUUIDるたびに新しい を作成します。この情報はレスポンスで DescribeServers API使用できます。

`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に変更されることはありません。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、レスポンスでも利用できます DescribeServers API。

`ServerId:`

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

`DnsNames:`

このオブジェクトに含まれる Amazon DCVサーバーに関連付けられたDNS名前を含むオブジェクト：

`InternalDnsNames:`

Amazon DCV Connection Gateway がインスタンスに接続するために使用するDNS名前。

Session Manager Broker CLI コマンドを使用して、ファイルからマッピングを`register-server-dns-mapping`ロードし (コマンドページ参照: register-server-dns-mapping) `describe-server-dns-mappings`、Session Manager Broker に現在ロードされているマッピングを一覧表示します (コマンドページ参照: 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。

Amazon DCV Session Manager のアップグレード

Amazon DCVシステムの規模と複雑さが増すにつれて、Session Manager が up-to-date引き続き需要の増加に対応できるようにすることが重要です。エージェントパッケージとブローカーパッケージの両方が、随時アップグレードする必要があります。このセクションでは、Amazon DCV Session Manager をアップグレードするプロセスの概要を説明し、システムを維持するためのアップグレード手順と推奨事項について説明します。

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**
>
> 新機能が導入されたときの非互換性の問題を避けるため、Session Manager ブローカーをアップグレードする前に、すべての Session Manager 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。

トピック

Amazon DCV Session Manager エージェントのアップグレード

Amazon DCV Session Manager エージェントはブローカーから指示を受け取り、それぞれのAmazon DCVサーバーで実行します。定期的なメンテナンスの一環として、エージェントは新しい標準と要件を満たすようにアップグレードする必要があります。このセクションでは、Session Manager エージェントのアップグレードプロセスについて説明します。

Linux host

> **Note**
>
> 以下の手順は、64 ビット x86 ホストにエージェントをインストールするためのものです。64 ビットARMホストにエージェントをインストールするには、Amazon Linux、、RHELおよび Centos の場合は、*x86_64* を にaarch64、Ubuntu の場合は を置き換えます。*amd64* arm64 で。

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 および RHEL 7.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent-2024.0.781-1.el7.x86_64.rpm
```

   - RHEL 8.x および Rocky Linux 8.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent-2024.0.781-1.el8.x86_64.rpm
```

   - Ubuntu 20.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent_2024.0.781-1_amd64.ubuntu2004.deb
```

   - Ubuntu 22.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent_2024.0.781-1_amd64.ubuntu2204.deb
```

   - Ubuntu 24.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent_2024.0.781-1_amd64.ubuntu2404.deb
```

   - SUSE Linux Enterprise 12

```
$  curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent-2024.0.781-1.sles12.x86_64.rpm
```

   - SUSE Linux Enterprise 15

```
$  curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerAgents/
nice-dcv-session-manager-agent-2024.0.781-1.sles15.x86_64.rpm
```

3. パッケージをインストールします。

   - Amazon Linux 2 および RHEL 7.x

   ```
   $  sudo yum install -y nice-dcv-session-manager-
   agent-2024.0.781-1.el7.x86_64.rpm
   ```

   - RHEL 8.x および Rocky Linux 8.x

   ```
   $  sudo yum install -y nice-dcv-session-manager-
   agent-2024.0.781-1.el8.x86_64.rpm
   ```

   - Ubuntu 20.04

   ```
   $  sudo apt install ./nice-dcv-session-manager-
   agent_2024.0.781-1_amd64.ubuntu2004.deb
   ```

   - Ubuntu 22.04

   ```
   $  sudo apt install ./nice-dcv-session-manager-
   agent_2024.0.781-1_amd64.ubuntu2204.deb
   ```

   - Ubuntu 24.04

   ```
   $  sudo apt install ./nice-dcv-session-manager-
   agent_2024.0.781-1_amd64.ubuntu2404.deb
   ```

   - SUSE Linux Enterprise 12

   ```
   $  sudo zypper install nice-dcv-session-manager-
   agent-2024.0.781-1.sles12.x86_64.rpm
   ```

   - SUSE Linux Enterprise 15

   ```
   $  sudo zypper install nice-dcv-session-manager-
   agent-2024.0.781-1.sles15.x86_64.rpm
   ```

4. 次のコマンドを実行してエージェントを起動します。

```
$  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 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 画面でライセンス契約を注意深く読み、同意する場合は、条件に同意し、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
```

Amazon DCV Session Manager ブローカーのアップグレード

Amazon DCV Session Manager ブローカーは、関連するエージェントにAPIリクエストを渡します。これらは Amazon DCVサーバーとは別のホストにインストールされます。定期的なメンテナンスの一環として、ブローカーは新しい標準と要件を満たすためにアップグレードする必要があります。このセクションでは、Session Manager ブローカーのアップグレードプロセスについて説明します。

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 および RHEL 7.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerBrokers/nice-
dcv-session-manager-broker-2024.0.457-1.el7.noarch.rpm
```

- RHEL 8.x および Rocky Linux 8.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerBrokers/nice-
dcv-session-manager-broker-2024.0.457-1.el8.noarch.rpm
```

- Ubuntu 20.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerBrokers/nice-
dcv-session-manager-broker-2024.0.457-1_all.ubuntu2004.deb
```

- Ubuntu 22.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerBrokers/nice-
dcv-session-manager-broker-2024.0.457-1_all.ubuntu2204.deb
```

- Ubuntu 24.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2024.0/SessionManagerBrokers/nice-
dcv-session-manager-broker-2024.0.457-1_all.ubuntu2404.deb
```

4. パッケージをインストールします。

- Amazon Linux 2 および RHEL 7.x

```
$  sudo yum install -y nice-dcv-session-manager-
broker-2024.0.457-1.el7.noarch.rpm
```

- RHEL 8.x および Rocky Linux 8.x

```
$  sudo yum install -y nice-dcv-session-manager-
broker-2024.0.457-1.el8.noarch.rpm
```

- Ubuntu 20.04

```
$  sudo apt install -y nice-dcv-session-manager-
broker-2024.0.457-1_all.ubuntu2004.deb
```

- Ubuntu 22.04

```
$  sudo apt install -y nice-dcv-session-manager-
broker-2024.0.457-1_all.ubuntu2204.deb
```

- Ubuntu 24.04

```
$  sudo apt install -y nice-dcv-session-manager-
broker-2024.0.457-1_all.ubuntu2404.deb
```

5. ブローカーサービスを起動し、インスタンスが起動するたびに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

Amazon DCV Session Manager ブローカーは、Session Manager を管理するコマンドラインインターフェイス (CLI) ツールです。このリファレンスでは、セッション、ユーザー、リソース、および Session Manager のその他の側面の管理に使用できるCLIコマンドの完全なセットについて説明します。管理者は、日常的な管理タスクを自動化し、問題のトラブルシューティングを行い、Amazon DCVインフラストラクチャのパフォーマンスを最適化できます。

外部認証サーバーを使用して OAuth2.0 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

Session Manager ブローカーを 2.0 OAuth 認証サーバーとして使用する場合は、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- キー

次のコマンドを使用して、Session Manager 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

次のコマンドを使用して、DCVサーバーDNSの名前マッピングファイルを管理します。

- 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、Session Manager はブローカーを認証サーバーとして使用して 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は、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

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

型: 文字列

必須: はい

例

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

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

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

出力

`Urls`

登録URLsされた外部認証サーバーの 。

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。外部認証サーバーを登録解除すると、OAuth2.0 アクセストークンの生成に使用することはできません。

トピック

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

型: 文字列

必須: はい

出力

`Url`

未登録URLの外部認証サーバーの 。

例

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

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

Session Manager クライアントをブローカーに登録し、クライアントがAPIリクエストを行うために必要な OAuth 2.0 アクセストークンを取得するために使用できるクライアント認証情報を生成します。

> **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 2.0 OAuth 認証サーバーとして使用されている場合にのみ使用されます。

トピック

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

Session Manager クライアントが OAuth2.0 アクセストークンを取得するために使用する一意のクライアント ID。

client-password

Session Manager OAuth クライアントが 2.0 アクセストークンを取得するために使用するパスワード。

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た Session Manager クライアントを一覧表示します。

トピック

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 Session Manager クライアントは、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

```
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- キー

ブローカーが Session Manager クライアントに提供された OAuth 2.0 アクセストークンに署名するために使用す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リクエストを行うために必要な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

Syntax

```
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。エージェントは、ブローカーに登録するためにソフトウェアステートメントが必要です。エージェントは、ソフトウェ

アステートメントを取得した後、OAuth2.0 動的クライアント登録プロトコル を使用して自動的にブローカーに登録できます。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は、最初にインストールされたときにデフォルトのソフトウェアステートメントを受信して使用します。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、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、ファイル内の `agent.software_statement_path`パラメータに`agent.conf`ファイルパスを追加する必要があります。これが完了したら、エージェントを停止して再起動し、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るようにします。

トピック

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

```
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を非アクティブ化すると、エージェント登録に使用することはできません。

トピック

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について説明します。

トピック

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true場合は です。それ以外の場合は です。false。

例

次の例では、 エージェントについて説明します。

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

ブローカーからエージェントを登録解除します。

トピック

- 構文

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、エージェントを登録解除します。

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

DCV サーバーを登録する - JSON ファイルから取得されるDNS名前マッピング。

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 Servers を含むファイルのパス - DNS名前マッピング。

型: 文字列

必須: はい

例

次の例では、/tmp/mappings.json ファイルのDCVサーバー - DNS名前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 - DNS名前マッピングについて説明します。

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

次の例では、登録されたDCVサーバーDNSの名前マッピングを一覧表示します。

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、Session Manager で使用可能な設定オプションの概要を示します。設定には、エージェントファイルとブローカーファイルの両方に対する変更が含まれます。各設定には、目的、許容値、システム動作全体への影響の説明が含まれます。Amazon DCV Session Manager は、Amazon DCVシステムの固有の要件を満たすようにカスタマイズできます。

トピック

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、Session Manager 機能をカスタマイズするよう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と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。ブローカーは、起動時に、ファイルに期待されるアクセス許可があることを確認します。

次の表に、ブローカー設定ファイルのパラメータを示します。

パラメータ名	必須	デフォルト値	説明
`broker.ja`	不可		システムのデフォルトディレクトリの代わりに、ブローカーが使用する Java

パラメータ名	必須	デフォルト値	説明
va-home			ホームディレクトリへのパスを指定します。設定されている場合、ブローカーは起動<broker-java-home>/bin/java 時にを使用します。 ヒント: ブローカーには Java Runtime Environment 11 が必要であり、インストールが成功したときに依存関係として欠落している場合は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 creensh-max-width	不可	160	を使用して撮影されたセッションスクリーンショットの最大幅をピクセル単位で指定しますGetSessionScreenshotsAPI。

パラメータ名	必須	デフォルト値	説明
sessions-screenshot-max-height	不可	100	を使用して撮影されたセッションスクリーンショットの最大高さをピクセル単位で指定しますGetSessionScreenshotsAPI。
sessions-screenshot-format	不可	png	を使用して撮影されたセッションスクリーンショットの画像ファイル形式GetSessionScreenshotsAPI。
create-sessions-queue-max-size	不可	1000	キューに入れることができる未完了のCreateSessionsAPIリクエストの最大数。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	不可	1800	未完了のCreateSessionsAPIリクエストがキューに残る最大時間を秒単位で指定します。リクエストは、指定時間内に処理されないと失敗します。

パラメータ名	必須	デフォルト値	説明
session manager-working-path	可能	/tmp	ブローカーが操作に必要なファイルを書き込む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、ブローカーのみがアクセスできます。
enable-authorization-server	可能	true	ブローカーがクライアントの 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かどうかを指定しますAPIs。
enable-authorization	可能	true	クライアント認証を有効または無効にします。クライアント認証を有効にする場合、クライアントはAPIリクエストを行うときにアクセストークンを提供するAPI必要があります。クライアント認証を無効にすると、クライアントAPIsはアクセストークンなしでリクエストを行うことができます。

パラメータ名	必須	デフォルト値	説明
enable-agent-authorizatic	可能	true	エージェント認証を有効または無効にします。エージェント認証を有効にする場合、エージェントはブローカーと通信するときにアクセストークンを提供する必要があります。
delete-session-duration-hours	不可	1	削除されたセッションが非表示になり、DescribeSession API呼び出しによって返されなくなった時間数を指定します。
connect session-token-duration-minutes	不可	60	ConnectSession トークンが有効のままである分数を指定します。
client-to-broker-connecto https-port	可能	8443	ブローカーがクライアント接続をリッスンするHTTPSポートを指定します。

パラメータ名	必須	デフォルト値	説明
client-to-broker-connect bind-host	不可	0.0.0.0	ブローカーがクライアント接続のためにバインドするホストの IP アドレスを指定します。
client-to-broker-connect key-store-file	可能		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-connect key-store-pass	可能		キーストアのパスを指定します。

パラメータ名	必須	デフォルト値	説明
agent-to-broker-connector-https-port	可能	8445	ブローカーがエージェント接続をリッスンするHTTPSポートを指定します。
agent-to-broker-connector-bind-host	不可	0.0.0.0	エージェント接続のためにブローカーがバインドするホストの IP アドレスを指定します。
agent-to-broker-connector-key-store-file	可能		TLS エージェント接続に使用されるキーストアを指定します。

パラメータ名	必須	デフォルト値	説明
agent-to-broker-connector-key-store-pass	可能		キーストアのパスを指定します。
broker-to-broker-port	可能	47100	接続に使用される broker-to-brokerポートを指定します。
broker-to-broker-bind-host	不可	0.0.0.0	ブローカーが接続のためにbroker-to-brokerバインドするホストの IP アドレスを指定します。
broker-to-broker-discover-port	可能	47500	ブローカーが相互に検出するために使用するポートを指定します。

パラメータ名	必須	デフォルト値	説明
broker-to-broker-discover-address	不可		のフリート内の他のブローカーの IP アドレスとポートを指定します。*ip_address*:*port* の形式で設定。複数のブローカーがある場合は、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。

パラメータ名	必須	デフォルト値	説明
broker-to-broker-discovery-multicast-group	不可		検出する broker-to-roker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	不可		検出する broker-to-broker 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。

パラメータ名	必須	デフォルト値	説明
broker-to-broker-discovery-AWS-region	不可		ブローカーからブローカーの検出に使用されるアプリケーションロードバランサーの 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	不可		検出用のARN broker-to-brokerアプリケーションロードバランサーターゲットグループユーザーの 。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。

パラメータ名	必須	デフォルト値	説明
broker-to-broker-distributed-memory-max-size-mb	不可	4096	Amazon DCVセッションデータを保存するために単一のブローカーが使用す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	可能		TLS ブローカー接続に使用されるキーストアを指定します。
broker-to-broker-key-store-pass	可能		キーストアのパスを指定します。
enable-cloud-watch-metrics	不可	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する場合は、 の値を指定する必要がありますcloud-watch-region 。

パラメータ名	必須	デフォルト値	説明
cloud-watch-region	不可	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 から取得されますIMDS。	CloudWatch メトリクスが投稿される AWS リージョン。
max-api-requests-per-秒	不可	1000	ブローカー API がスロットリングする前に 1 秒ごとに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	不可	false	スロットリングに設定するtrueと、存在する場合は、ヘッダーから発信者 X-Forwared-Forip を取得します。
create-sessions-number-of-retries-on-failure	不可	2	Amazon 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後に実行される再試行の最大数を指定します。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。

パラメータ名	必須	デフォルト値	説明
autorun file-arguments-max-size	不可	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。
autorun file-arguments-max-argumen length	不可	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	可能	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	不可	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	不可	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。

パラメータ名	必須	デフォルト値	説明
dynamoc table-rcu	不可	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ユニット (RCU) を指定します。 の詳細についてはRCU、「プロビジョンドキャパシティの料金」を参照してください。
dynamoc table-wcu	不可	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。の詳細についてはWCU、「 プロビジョンドキャパシティの料金」を参照してください。
dynamoc table-name-prefix	不可	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 テーブルに追加す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区別するのに役立ちます)。英数字、ドット、ダッシュ、下線のみを使用できます。

パラメータ名	必須	デフォルト値	説明
jdbc-connection-url	不可	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URLへの接続を指定します。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` ここで、 <db_endpoint> は MariaDB /MySQL 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	不可	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	不可	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。

パラメータ名	必須	デフォルト値	説明
seconds_before-deleting-unreachable-dcv-server	不可	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 (/etc/dcv-session-manager-agent/agent.confLinux および Windows C:\Program Files\NICE\DCVSessionManagerAgent\conf\agent.conf 用) には、Session Manager 機能をカスタマイズするよう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。

次の表に、エージェント設定ファイルのパラメータを示します。

パラメータ名	必須	デフォルト値	[Description] (説明)
agent.broker_host	可能		ブローカーホストDNSの名前を指定します。
agent.broker_port	可能	8445	ブローカーと通信するポートを指定します。
agent.ca_file	不可		tls_strict が true に設定されている場合のみ必要です。証明書の検証

パラメータ名	必須	デフォルト値	[Description] (説明)
			に必要なTLS証明書 (.pem) ファイル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ます。
agent.i t_folde	不可	• /var/lib/dcv-session-manager-agent/init (Linux)	Amazon サーバーセッションの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されるホストDCVサーバー上のフォルダ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。フォルダはアクセス可能で、ファイルは のInitFileリクエストパラメータを使用するユーザーが実行可能である必要がありますCreateSessionsAPI。
agent.t _strict	不可	true	厳密なTLS検証を使用するかどうかを示します。
agent.s tware_s tement_ th	不可		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。

パラメータ名	必須	デフォルト値	[Description] (説明)
agent.t s_folde	不可	• /etc/dcv-session-manager-agent (Linux) • C:\Program Files\NICE\DCVSessionManagerAgent\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、「タグを使用して Amazon DCVサーバーをターゲットにする」を参照してください。
agent.a orun_fo er	不可	• /var/lib/dcv-session-manager-agent/autorun (Linux) • C:\ProgramData\NICE\DcvSessionManagerAgent\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。フォルダはアクセス可能で、ファイルは のAutorunFileリクエストパラメータを使用するユーザーが実行可能である必要がありますCreateSessionsAPI。
agent.m _virtua sessior	不可	-1 (制限なし)	Amazon Session Manager を使用して Amazon DCV サーバーで作成できる仮想DCVセッションの最大数。

パラメータ名	必須	デフォルト値	[Description] (説明)
agent.max_concurrent_sessions_per_user	不可	1	Amazon Session Manager を使用して、1 人のユーザーが Amazon DCVサーバーで作成できる仮想DCVセッションの最大数。
agent.broker_update_interval	不可	30	ブローカーに更新されたデータを送信するまで待機する秒数を指定します。送信データには、Amazon 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。値が低いと、Session Manager はエージェントが実行されるシステムで起こっている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。

パラメータ名	必須	デフォルト値	[Description] (説明)
log.lev	不可	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.dir tory	不可	• /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。

パラメータ名	必須	デフォルト値	[Description] (説明)
log.rot ion	不可	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.ma> f ile- size	不可	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.rot e	不可	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。

Amazon DCV Session Manager のリリースノートとドキュメント履歴

このページでは、Amazon DCV Session Manager のリリースノートとドキュメント履歴について説明します。

トピック

Amazon DCV Session Manager リリースノート

このセクションでは、Amazon DCV Session Manager の主要な更新、機能リリース、バグ修正の概要を説明します。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

- 2020.2-9662 — 2020 年 12 月 4 日
- 2020.2-9508 — 2020 年 11 月 11 日

2024.0 ~ 457 - 2024 年 10 月 1 日

ビルド番号	変更とバグ修正
• ブローカー: 457 • エージェント: 748 • CLI: 140	• Amazon NICEDCVにブランド変更されましたDCV。 • Ubuntu 24.04 のサポートを追加しました。

2023.1 ~ 17652 - 2024 年 8 月 1 日

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

2023.1 ~ 16388 - 2024 年 6 月 26 日

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

2023.1— 2023 年 11 月 9 日

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

ビルド番号	変更とバグ修正
• エージェント: 732 • CLI: 140	

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の Red Hat Enterprise Linux 9、Rocky Linux 9、および CentOS Stream 9 のサポートが追加されました。

2023.0-14852— 2023 年 3 月 28 日

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

2022.2-13907— 2022 年 11 月 11 日

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

2021.3-11591 — 2021 年 12 月 20 日

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

2021.2-11445 — 2021 年 11 月 18 日

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

2021.2-11190 — 2021 年 10 月 11 日

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

2021.2-11042 — 2021 年 9 月 1 日

ビルド番号	新機能	変更とバグ修正
• ブローカー: 254 • エージェント: 413 • CLI: 37	• Amazon DCV Session Manager がコマンドラインインターフェイス (CLI) のサポートを提供するようになりました。を呼び出す代わりにCLI、 でAmazon DCVセッションを作成および管理できますAPIs。 • Amazon DCV Session Manager は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。	• 外部認証サーバーを登録するときに、認証サーバーが JSON形式のウェブトークンに署名するために使用するアルゴリズムを指定できるようになりました。この変更により、Azure AD を外部認可サーバーとして使用できます。

2021.1-10557 — 2021 年 5 月 31 日

ビルド番号	新機能	変更とバグ修正
• ブローカー: 214 • エージェント: 365	• Amazon DCV Session Manager は、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 に渡せるようになりましたAPI。	• Windows での自動実行ファイルに関する問題を修正しました。

2021.0-10242 — 2021 年 4 月 12 日

ビルド番号	変更とバグ修正

• ブローカー: 183
• エージェント: 318
	• Amazon DCV Session Manager では、次の新しい が導入されましたAPIs。
	  • OpenServers
	  • CloseServers
	  • DescribeServers
	  • GetSessionScreenshots
	• 次の新しい設定パラメータも導入しました。
	  • ブローカーパラメータ: `session-screenshot-max-width` 、`session-screenshot-max-height` 、`session-screenshot-format` 、`create-sessions-queue-max-size` 、`create-sessions-queue-max-time-seconds`
	  • 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`

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

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	• ブローカーの起動を妨げる自動生成されたTLS証明書の問題を修正しました。

2020.2-9508 — 2020 年 11 月 11 日

ビルド番号	変更とバグ修正
• ブローカー: 78 • エージェント: 183	• Amazon DCV Session Manager の初期リリース。

ドキュメント履歴

次の表は、Amazon DCV Session Manager のこのリリースのドキュメントを示しています。

変更	説明	日付
Amazon DCVバージョン 2024.0-457	Amazon DCV Session Manager が Amazon 2024.0-457 DCV 用に更新されました。詳細については、「2024.0 ~ 457 - 2024 年 10 月 1 日」を参照してください。	2024 年 9 月 30 日
Amazon DCVバー	Amazon DCV Session Manager が Amazon 2023.1-17652 DCV 用に更新さ	2024 年 8 月 1 日

変更	説明	日付
ジョン 2023.1-17652	れました。詳細については、「2023.1 ~ 17652 - 2024 年 8 月 1 日」を参照してください。	
Amazon DCVバージョン 2023.1-16388	Amazon DCV Session Manager が Amazon 2023.1 ~ 16388 DCV 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
Amazon DCVバージョン 2023.1	Amazon DCV Session Manager が Amazon 2023.1 DCV 用に更新されました。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
Amazon DCVバージョン 2023.0	Amazon DCV Session Manager が Amazon 2023.0 DCV 用に更新されました。詳細については、「2023.0-14852 — 2023 年 3 月 28 日」を参照してください。	2023 年 3 月 28 日
Amazon DCVバージョン 2022.2	Amazon DCV Session Manager が Amazon 2022.2 DCV 用に更新されました。詳細については、「2022.2-13907 — 2022 年 11 月 11 日」を参照してください。	2022 年 11 月 11 日
Amazon DCVバージョン 2022.1	Amazon DCV Session Manager が Amazon 2022.1 DCV 用に更新されました。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

変更	説明	日付
Amazon DCVバージョン 2022.0	Amazon DCV Session Manager が Amazon 2022.0 DCV 用に更新されました。詳細については、「2022.0-11952 — 2022 年 2 月 23 日」を参照してください。	2022 年 2 月 23 日
Amazon DCVバージョン 2021.3	Amazon DCV Session Manager が Amazon 2021.3 DCV 用に更新されました。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
Amazon DCVバージョン 2021.2	Amazon DCV Session Manager が Amazon 2021.2 DCV 用に更新されました。詳細については、「2021.2-11042 — 2021 年 9 月 1 日」を参照してください。	2021 年 9 月 1 日
Amazon DCVバージョン 2021.1	Amazon DCV Session Manager が Amazon 2021.1 DCV 用に更新されました。詳細については、「2021.1-10557 — 2021 年 5 月 31 日」を参照してください。	2021 年 5 月 31 日
Amazon DCVバージョン 2021.0	Amazon DCV Session Manager が Amazon 2021.0 DCV 用に更新されました。詳細については、「2021.0-10242 — 2021 年 4 月 12 日」を参照してください。	2021 年 4 月 12 日
Amazon DCV Session Manager 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。