NICE DCV 会话管理器

管理员指南

亚马逊云科技

NICE DCV 会话管理器: 管理员指南

Table of Contents

什么是 NICE DCV 会话管理器？

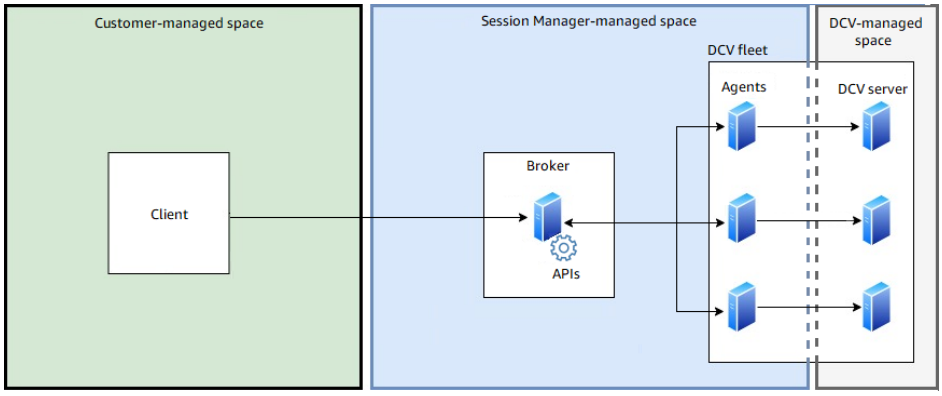
NICE DCV Session Manager 是一套可安装的软件包（代理和经纪商）和应用程序编程接口 (API)，使开发人员和独立软件供应商 (ISV) 能够轻松构建以编程方式创建和管理 NICE DCV 生命周期的前端应用程序跨 NICE DCV 服务器队列的会话。

此指南介绍如何安装和配置会话管理器代理和代理。有关使用会话管理器 API 的更多信息，请参阅NICE DCV 会话管理器开发者指南.

主题

会话管理器如何工作

下图显示会话管理器的高级组件。



代理

经纪商是托管和公开会话管理器 API 的 Web 服务器。它接收和处理API请求管理 NICE DCV 会话客户，然后将说明传递给相关人员客服. 代理必须安装在独立于 NICE DCV 服务器的主机上，但客户端必须可以访问该代理，并且必须能够访问代理。

代理

代理程序安装在队列中的每台 NICE DCV 服务器上。代理接收来自代理然后在各自的 NICE DCV 服务器上运行它们。代理还监控 NICE DCV 服务器的状态，并将定期更新状态发送回经纪商。

API

会话管理器公开了一组 REST 应用程序编程接口 (API)，可用于管理 NICE DCV 服务器队列上的 NICE DCV 会话。API 托管并由代理. 开发人员可以构建自定义会话客户那叫 API。

客户端

客户端是您开发的调用会话管理器的前端应用程序或门户API被暴露的代理. 最终用户使用客户端管理队列中 NICE DCV 服务器上托管的会话。

访问令牌

要发出 API 请求，您必须提供访问令牌。可以通过注册的客户端 API 从经纪商或外部授权服务器请求令牌。要请求和访问令牌，客户端 API 必须提供有效的凭据。

客户端 API

客户端 API 是使用 Swagger Codegen 从会话管理器 API 定义 YAML 文件生成的。客户端 API 用于发出 API 请求。

NICE DCV 会话

您必须在 NICE DCV 服务器上创建一个可以连接到的 NICE DCV 会话。如果有活动会话，客户端只能连接到 NICE DCV 服务器。NICE DCV 支持控制台和虚拟会话。您可以使用会话管理器 API 来管理 NICE DCV 会话的生命周期。NICE DCV 会话可以处于以下状态之一：

- `CREATING`— 代理正在创建会话。
- `READY`— 会话已准备好接受客户端连接。
- `DELETING`— 会话正在删除。
- `DELETED`— 会话已删除。
- `UNKNOWN`— 无法确定会话的状态。经纪人和代理可能无法进行沟通。

功能

DCV 会话管理器提供以下功能：

- 提供 NICE DCV 会话信息— 获取有关在多个 NICE DCV 服务器上运行的会话的信息。
- 管理多个 NICE DCV 会话的生命周期— 通过一个 API 请求跨多个 NICE DCV 服务器为多个用户创建或删除多个会话。
- 支持标签— 在创建会话时，使用自定义标签来定位一组 NICE DCV 服务器。
- 管理多个 NICE DCV 会话的权限— 使用一个 API 请求修改多个会话的用户权限。
- 提供连接信息— 检索 NICE DCV 会话的客户端连接信息。
- 支持云和本地— 在上使用会话管理器Amazon、本地或使用其他基于云的服务器。

限制

会话管理器不提供资源调配功能。如果您在 Amazon EC2 实例上运行 NICE DCV，则可能需要使用额外的Amazon服务，例如 Amazon EC2 Auto Scaling，用于管理基础设施的扩展。

定价

会话管理器免费使用Amazon运行 EC2 实例的客户。

本地客户需要 NICE DCV Plus 或 DCV 专业 Plus 许可证。有关如何购买 NICE DCV Plus 或 NICE DCV 专业 Plus 许可证的信息，请参阅如何购买在 NICE 网站上找到您所在地区的 NICE 分销商或经销商。为了允许所有本地客户试用 DCV 会话管理器，只能从 NICE DCV 版本 2021.0 开始执行许可要求。

有关更多信息，请参阅 。许可 NICE DCV 服务器中的NICE DCV 管理员指南.

NICE DCV 会话管理器要求

NICE DCV 会话管理器代理和经纪商有以下要求。

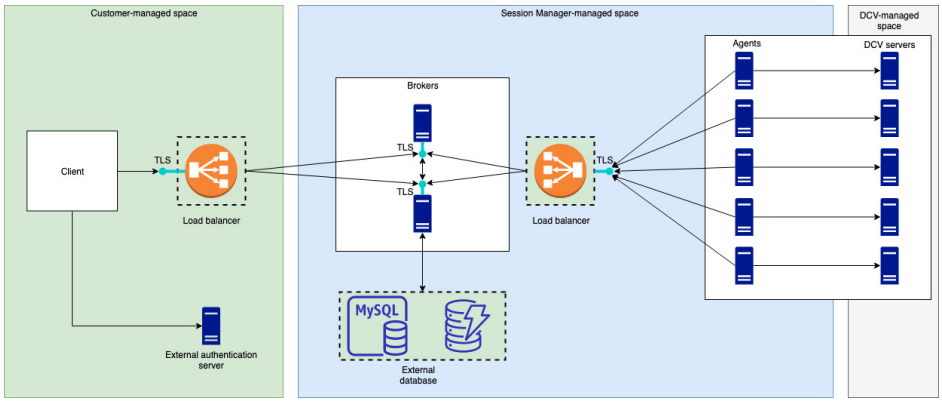
	代理	代理

操作系统	• Amazon Linux 2
	• CentOS 7.6 或更高版本
	• CentOS 8.x
	• RHEL 7.6 或更高版本
	• RHEL 8.x
	• Ubuntu 18.04
	• Ubuntu 20.04
		• Windows
		  • Windows Server 2019
		  • Windows Server 2016
		  • Windows Server 2012 R2
		• Linux 服务器
		  • Amazon Linux 2
		  • CentOS 7.6 或更高版本
		  • CentOS 8.x
		  • RHEL 7.6 或更高版本
		  • RHEL 8.x
		  • Ubuntu 18.04
		  • Ubuntu 20.04
		  • SUSE Linux Enterprise 12（带 SP3）或更高版本
		  • SUSE Linux Enterprise 15

架构	64 位 x86	• 64 位 x86 • 64 位 ARM（仅限亚马逊 Linux 2、CentOS 7.x、RHEL 7.x）和 Ubuntu 18.04
内存	8GB	4 GB
NICE DCV 版本	不错的 DCV 2020.2 及更高版本	不错的 DCV 2020.2 及更高版本
其他要求	Java 11	-

网络和连接要求

下图高度概述了会话管理器的网络和连接要求。



这些区域有：代理必须安装在单独的主机上，但它必须与 NICE DCV 服务器上的 Agent 进行网络连接。如果您选择有多个经纪商来提高可用性，则必须在单独的主机上安装和配置每个经纪商，并使用一个或多个负载均衡器来管理客户与经纪商、经纪商和代理之间的流量。经纪商还应该能够彼此沟通，以便交换有关 NICE DCV 服务器和会话的信息。经纪商可以将其密钥和状态数据存储在外部数据库中，并在重启或终止后获得这些信息。通过将重要的经纪商信息保存在外部数据库中，这有助于降低丢失重要的经纪商信息 您可以稍后检索它。如果您选择拥有它，则必须设置外部数据库并配置代理。支持 DynamoDB、MariaDB 和 MySQL。您可在代理配置文件.

这些区域有：客服必须能够与经纪商启动安全、持久、双向 HTTPS 连接。

您的客户或前端应用程序，必须能够访问经纪商才能调用 API。客户端还应能访问您的身份验证服务器。

设置 NICE DCV 会话管理器

以下部分介绍了如何使用单个 Broker 和多个代理程序安装会话管理器。您可以使用多个经纪商来提高可扩展性和性能。有关更多信息，请参阅 扩展会话管理器 (p. 13)。

要设置 NICE DCV 会话管理器，请执行以下操作：

步骤

第 1 步：准备好好 DCV 服务器

你必须拥有一组 NICE DCV 服务器，你打算使用会话管理器。有关安装 NICE DCV 服务器的更多信息，请参阅安装 NICE DCV 服务器中的NICE DCV 管理员指南.

在 Linux NICE DCV 服务器上，会话管理器使用名为`dcvsmagent`. 安装会话管理器代理时会自动创建此用户。您必须为 NICE DCV 授予此服务用户管理员权限，以便它可以代表其他用户执行操作。要授予会话管理器服务用户管理员权限，请执行以下操作：

为 Linux NICE DCV 服务器添加本地服务用户

1. 使用所需的文本编辑器打开 `/etc/dcv/dcv.conf`。
2. 添加`administrators`参数`[security]`部分，然后指定会话管理器用户。例如：

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

3. 保存并关闭 文件。
4. 停止并重新启动 NICE DCV 服务器。

会话管理器只能代表 NICE DCV 服务器上已存在的用户创建 NICE DCV 会话。如果请求为不存在的用户创建会话，则该请求将失败。因此，您必须确保每个预期最终用户在 NICE DCV 服务器上都有一个有效的系统用户。

Tip

如果您打算将多个 Broker 主机或 NICE DCV 服务器与代理结合使用，我们建议您只配置一个 Broker 和一个 NICE DCV 服务器，方法是执行以下步骤，创建具有已完成配置的主机的 Amazon 系统映像 (AMI)，然后使用 AMI启动剩余的代理和 NICE DCV 服务器。此外，也可以使用 AmazonSystems Manager 可在多个实例上远程运行命令。

第 2 步：设置 NICE DCV 会话管理器经纪商

代理必须安装在 Linux 主机上。有关支持的 Linux 发行版的更多信息，请参阅NICE DCV 会话管理器要求 (p. 3). 在独立于代理和 NICE DCV 服务器主机的主机上安装 Broker。主机可以安装在另一个专用网络上，但必须能够连接到代理并与 Agent 进行通信。

安装和启动代理

1. Connect 到您打算在其上安装经纪商的主机。
2. 程序包使用安全 GPG 签名进行数字签名。要允许程序包管理器验证程序包签名，您必须导入 NICE GPG 密钥。运行以下命令以导入 NICE GPG 密钥。

   - Amazon Linux 2、RHEL 7.x、RHEL 8.x 和 CentOS 7.x 和 CentOS 8.x

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

   - Ubuntu 18.04 和 Ubuntu 20.04

   ```
   $ sudo apt-key add https://d1uj6qtbmh3dt5.cloudfront.net/NICE-GPG-KEY
   ```

3. 下载安装包。

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

   ```
   $ wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerBrokers/nice-dcv-
   session-manager-broker-2022.0.341-1.el7.noarch.rpm
   ```

   - RHEL 8.x 和 CentOS 8.x

   ```
   $ wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerBrokers/nice-dcv-
   session-manager-broker-2022.0.341-1.el8.noarch.rpm
   ```

   - Ubuntu 18.04

   ```
   $ wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerBrokers/nice-dcv-
   session-manager-broker-2022.0.341-1_all.ubuntu1804.deb
   ```

   - Ubuntu 20.04

   ```
   $ wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerBrokers/nice-dcv-
   session-manager-broker-2022.0.341-1_all.ubuntu2004.deb
   ```

4. 安装 软件包。

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

   ```
   $ sudo yum install -y nice-dcv-session-manager-broker-2022.0.341-1.el7.noarch.rpm
   ```

   - RHEL 8.x 和 CentOS 8.x

   ```
   $ sudo yum install -y nice-dcv-session-manager-broker-2022.0.341-1.el8.noarch.rpm
   ```

   - Ubuntu 18.04

   ```
   $ sudo apt install -y nice-dcv-session-manager-broker-2022.0.341-1_all.ubuntu1804.deb
   ```

   - Ubuntu 20.04

   ```
   $ sudo apt install -y nice-dcv-session-manager-broker-2022.0.341-1_all.ubuntu2004.deb
   ```

5. 检查默认 Java 环境版本是否为11

   ```
   $ java -version
   ```

如果没有，你可以明确设置代理将用来定位正确的 Java 版本的 Java 主目录。设置参数完成`broker-java-home`在 Broker 配置文件中。有关更多信息，请参阅 。代理配置文件.

6. 启动 Broker 服务并确保它在每次启动实例时自动启动。

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

7. 将经纪商的自签名证书的副本放在您的用户目录中。在下一步中安装客服时，您将需要使用该值。

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

第 3 步：设置 NICE DCV 会话管理器代理

该代理必须安装在队列中的所有 NICE DCV 服务器主机上。Windows 和 Linux 服务器上都可以安装代理。有关支持的操作系统的更多信息，请参阅NICE DCV 会话管理器要求 (p. 3).

先决条件

安装代理之前，必须在主机上安装 NICE DCV 服务器。

Linux host

Note

会话管理器代理可用于以下 Linux 发行版和体系结构：

- Amazon Linux 2（64 位 x86 和 64 位 ARM）
- RHEL 7.x 和 CentOS 7.x（64 位 x86 和 64 位 ARM）
- RHEL 8.x 和 CentOS 8.x（64 位 x86 和 64 位 ARM）
- Ubuntu 18.04 和 Ubuntu 20.04（64 位 x86 和 64 位 ARM）
- SUSE Linux 企业版 12 和 SUSE Linux 企业版 15（仅限 64 位 x86）

以下说明用于在 64 位 x86 主机上安装代理。要在 64 位 ARM 主机上安装代理，对于亚马逊 Linux、RHEL 和 Centos，请替换*`x86_64`*和`aarch64`，对于 Ubuntu，替换*`amd64`*和`arm64`.

在 Linux 主机上安装代理

1. 程序包使用安全 GPG 签名进行数字签名。要允许程序包管理器验证程序包签名，您必须导入 NICE GPG 密钥。运行以下命令以导入 NICE GPG 密钥。

   - Amazon Linux 2、RHEL、CentOS 和 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. 下载安装包。

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

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/nice-dcv-session-manager-agent-2022.0.520-1.el7.x86_64.rpm
```

- RHEL 8.x 和 CentOS 8.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/nice-dcv-session-manager-agent-2022.0.520-1.el8.x86_64.rpm
```

- Ubuntu 18.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/nice-dcv-session-manager-agent_2022.0.520-1_amd64.ubuntu1804.deb
```

- Ubuntu 20.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/nice-dcv-session-manager-agent_2022.0.520-1_amd64.ubuntu2004.deb
```

- SUSE Linux Enterprise 12

```
$ curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/nice-dcv-session-manager-agent-2022.0.520-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$ curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/nice-dcv-session-manager-agent-2022.0.520-1.sles15.x86_64.rpm
```

3. 安装 软件包。

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

```
$  sudo yum install -y nice-dcv-session-manager-agent-2022.0.520-1.el7.x86_64.rpm
```

- RHEL 8.x 和 CentOS 8.x

```
$  sudo yum install -y nice-dcv-session-manager-agent-2022.0.520-1.el8.x86_64.rpm
```

- Ubuntu 18.04

```
$  sudo apt install ./nice-dcv-session-manager-agent_2022.0.520-1_amd64.ubuntu1804.deb
```

- Ubuntu 20.04

```
$  sudo apt install ./nice-dcv-session-manager-agent_2022.0.520-1_amd64.ubuntu2004.deb
```

- SUSE Linux Enterprise 12

```
$ sudo zypper install nice-dcv-session-manager-agent-2022.0.520-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$ sudo zypper install nice-dcv-session-manager-
agent-2022.0.520-1.sles15.x86_64.rpm
```

4. 将经纪商的自签名证书（您在上一步中复制的）的副本放在 `/etc/dcv-session-manager-agent/`代理上的目录。
5. 打开`/etc/dcv-session-manager-agent/agent.conf`使用首选文本编辑器，并执行以下操作。

   - 适用于`broker_host`中，指定安装了 Broker 的主机的 DNS 名称。

     Important

     如果代理在 Amazon EC2 实例上运行，则`broker_host`您必须指定实例的私有 IPv4 地址。
   - （可选）对于`broker_port`中，指定要与经纪商进行通信的端口。默认情况下，代理和经纪商通过端口进行通信`8445`. 只有在需要使用不同的端口时才更改此项。如果您确实进行了更改，请确保将 Broker 配置为使用相同的端口。
   - 适用于`ca_file`中，指定您在上一步中复制的证书文件的完整路径。例如：

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

     或者，如果要禁用 TLS 验证，请设置`tls_strict`到`false`.
6. 保存并关闭 文件。
7. 运行以下命令以启动代理。

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

Windows host

在 Windows 主机上安装客服

1. 下载代理安装.
2. 运行安装程序。在欢迎屏幕上，选择 Next。
3. 在 EULA 屏幕上，仔细阅读许可协议，如果您同意，请选择我接受这些条款然后选择下一步.
4. 要开始安装，请选择安装.
5. 将经纪商的自签名证书（您在上一步中复制的）的副本放在`C:\Program Files\NICE\DCVSessionManagerAgent\conf\`代理上的文件夹。
6. 打开`C:\Program Files\NICE\DCVSessionManagerAgent\conf\agent.conf`使用首选文本编辑器，然后执行以下操作：

   - 适用于`broker_host`中，指定安装了 Broker 的主机的 DNS 名称。

     Important

     如果代理在 Amazon EC2 实例上运行，则`broker_host`您必须指定实例的私有 IPv4 地址。
   - （可选）对于`broker_port`中，指定要与经纪商进行通信的端口。默认情况下，代理和经纪商通过端口进行通信`8445`. 只有在需要使用不同的端口时才更改此项。如果您确实进行了更改，请确保将 Broker 配置为使用相同的端口。
   - 适用于`ca_file`中，指定您在上一步中复制的证书文件的完整路径。例如：

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

或者，如果要禁用 TLS 验证，请设置`tls_strict`到`false`.

7. 保存并关闭 文件。
8. 停止并重新启动 Agent 服务以使更改生效。在命令提示符处运行以下命令。

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

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

第 4 步：将 NICE DCV 服务器配置为使用 Broker 作为身份验证服务器

将 NICE DCV 服务器配置为使用 Broker 作为验证客户端连接令牌的外部身份验证服务器。您还必须配置 NICE DCV 服务器以信任经纪商的自签名 CA。

Linux NICE DCV server

为 Linux NICE DCV 服务器添加本地服务用户

1. 使用所需的文本编辑器打开 `/etc/dcv/dcv.conf`。
2. 添加`ca-file`和`auth-token-verifier`的参数`[security]`部分。

   适用于`ca-file`中，指定您在上一步中复制到主机中的代理商的自签名 CA 的路径。

   适用于`auth-token-verifier`中，请按以下格式为经纪商上的令牌验证器指定 URL：`https://`*`broker_ip_or_dns`*`:`*`port`*`/agent/validate-authentication-token`. 指定用于经纪商与代理通信的端口，默认情况下为 8445。如果您在 Amazon EC2 实例上运行经纪商，则必须使用私有 DNS 或私有 IP 地址。

   例如

   ```
   [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. 保存并关闭 文件。
4. 停止并重新启动 NICE DCV 服务器。有关更多信息，请参阅 。停止漂亮的 DCV 服务器和启动 NICE DCV 服务器中的NICE DCV 管理员指南.

Windows NICE DCV server

在 Windows NICE DCV 服务器上

1. 打开 Windows 注册表编辑器，并导航到HKEY_USERS/S-1-5-18/Software/GSettings/com/nictware/dcv/安全性/键。
2. 打开ca-file参数。适用于Value data中，指定您在上一步中复制到主机中的代理商的自签名 CA 的路径。

   Note

   如果该参数不存在，则创建一个新的字符串参数并为其命名。`ca-file`.

3. 打开auth-token-verifier参数。适用于Value data中，请按以下格式为经纪商上的令牌验证器指定URL：`https://broker_ip_or_dns:port/agent/validate-authentication-token`. 指定用于经纪商与代理通信的端口，默认情况下为 8445。如果您在 Amazon EC2 实例上运行经纪商，则必须使用私有 DNS 或私有 IP 地址。

   Note

   如果该参数不存在，则创建一个新的字符串参数并为其命名。`auth-token-verifier`.
4. 选择确定，并关闭 Windows 注册表编辑器。
5. 停止并重新启动 NICE DCV 服务器。有关更多信息，请参阅 。停止漂亮的 DCV 服务器和启动 NICE DCV 服务器中的NICE DCV 管理员指南.

第 5 步：验证安装

主题

验证客服

安装代理和代理之后，请确保代理正在运行并且能够连接到代理。

Linux 代理主机

要运行的命令取决于版本。

- 自 2022.0 版本以来

  从 Agent 主机运行以下命令：

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

- 2022.0 之前的版本

  从 Agent 主机上，运行以下命令，并指定当前年、月份和日期。

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

  例如

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

如果代理正在运行并且能够连接到代理程序，则该命令应返回`success`.

如果命令返回不同的输出，请检查 Agent 日志文件以了解更多信息。这些日志文件位于此处： `/var/log/dcv-session-manager-agent/`.

Windows 客服主机

打开代理日志文件，该文件位于`C:\ProgramData\NICE\DCVSessionManagerAgent\log`.

如果日志文件中包含类似于下面的行，则代理正在运行并且能够连接到 Broker。

```
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}"
```

如果您的日志文件没有类似的行，请检查日志文件是否存在错误。

验证代理

安装 Broker 和 Agent 后，请确保您的 Broker 正在运行，并且可以从用户和前端应用程序访问它。

从应该能够到达 Broker 的计算机上，运行以下命令：

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

如果验证成功，经纪商将返回以下内容：

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

配置 NICE DCV 会话管理器

本部分介绍如何为会话管理器执行高级配置。

主题

扩展会话管理器

要实现高可用性和提高性能，您可以将 Session Manager 配置为使用多个代理和经纪商。如果您确实打算使用多个代理和经纪商，我们建议您只安装和配置一个代理和代理主机，从这些主机创建 Amazon Machers 映像 (AMI)，然后从 AMI 启动剩余的主机。

默认情况下，会话管理器支持使用多个代理，而无需任何其他配置。但是，如果您打算使用多个经纪商，则必须使用负载均衡器来平衡前端客户和经纪商之间以及经纪商与代理之间的流量。负载均衡器的设置和配置完全由您拥有和管理。

以下部分介绍如何将 Session Manager 配置为将多个主机与应用 Application Load Balancer 结合使用。

步骤

第 1 步：创建实例配置文件

您必须将实例配置文件附加到代理主机和代理主机，以授予他们使用 Elastic Load Balancing API 的权限。有关更多信息，请参见适用于 Linux 实例的 Amazon EC2 用户指南中的 Amazon EC2 的 IAM 角色。

创建实例配置文件

1. 创建Amazon Identity and Access Management(IAM) 角色，用于定义在实例配置文件中使用的权限。请使用以下策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
        "elasticloadbalancing:DescribeTargetHealth"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

有关更多信息，请参阅 。创建 IAM 角色中的IAM 用户指南.

2. 创建新的实例配置文件。有关更多信息，请参阅 。创建实例配置文件中的Amazon CLI命令参考.
3. 将 IAM 角色添加到实例配置文件。有关更多信息，请参阅 。将角色添加到实例配置文件中的Amazon CLI命令参考.
4. 将实例配置文件附加到 Broker 主机。有关更多信息，请参阅 。将 IAM 角色附加到实例中的适用于 Linux 实例的 Amazon EC2 用户指南.

第 2 步：为负载均衡器准备 SSL 证书

当您对负载均衡器侦听器使用 HTTPS 时，必须在负载均衡器上部署 SSL 证书。负载均衡器先使用此证书终止连接，然后解密来自客户端的请求，最后再将请求发送到目标。

准备 SSL 证书

1. 创建私有证书颁发机构 (CA)AmazonCertificate Manager Private Certificate 颁发机构 (ACM PCA)。有关更多信息，请参阅 。创建 CA 的步骤中的Amazon Certificate Manager Private Certificate AU.
2. 安装 CA。有关更多信息，请参阅 。安装根 CA 证书中的Amazon Certificate Manager Private Certificate AU.
3. 请求由 CA 签名的新私有证书。对于域名，请使用`*.region.elb.amazonaws.com`并指定您打算在其中创建负载均衡器的区域。有关更多信息，请参阅 。请求私有证书中的Amazon Certificate Manager Private Certificate AU.

第 3 步：创建代理应用程序负载均衡器

创建应用程序负载均衡器以平衡前端客户端和经纪商之间的流量。

创建负载均衡器

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。

   在导航窗格中，选择负载均衡器然后选择创建 Load Balancer. 对于负载均衡器类型，请选择应用程序负载均衡器.
2. 适用于第 1 步：配置 Load Balancer中，执行以下操作：

   a. 适用于名称，输入一个描述性的负载均衡器名称。
   b. 适用于Scheme，选择面向 Internet 的.
   c. 适用于Load Balancer 协议，选择HTTPS，以及Load Balancer 端口输入，`8443`.
   d. 适用于VPC中，选择要使用的 VPC，然后选择该 VPC 中的所有子网。

e. 选择下一步。

3. 适用于步骤 2: 配置安全设置中，执行以下操作：

   a. 适用于证书类型，选择从 ACM 中选择证书.
   b. 适用于证书名中，选择您之前请求的私有证书。
   c. 选择下一步。

4. 适用于步骤 3: 配置安全组，创建一个新的安全组，或者选择一个现有的安全组，允许前端客户端与经纪商之间通过 HTTPS 和端口 8443 进行入站和出站流量。

   选择下一步。

5. 适用于步骤 4: 配置路由中，执行以下操作：

   a. 适用于目标组，选择新的目标组.
   b. 对于名称，输入目标组的名称。
   c. 适用于Target type，选择实例.
   d. 适用于协议，选择HTTPS. 对于端口，输入 `8443`。适用于协议版本，选择HTTP1.
   e. 对于运行状况检查协议，选择 HTTPS，然后Path (路径)输入，`/health`.
   f. 选择下一步。

6. 适用于第 5 步：注册目标，选择下一步.
7. 选择 Create (创建)。

第 4 步：启动代理

创建初始代理并将其配置为使用负载均衡器，从代理创建 AMI，然后使用 AMI 启动其余的经纪商。这可确保所有经纪商都配置为使用相同的 CA 和相同的负载均衡器配置。

启动经纪商

1. 启动并配置初始的 Broker 主机。有关安装和配置 Broker 的更多信息，请参阅第 2 步：设置 NICE DCV 会话管理器经纪商 (p. 5).
2. 打开 Connect 到代理`/etc/dcv-session-manager-broker/session-manager-broker.properties`使用首选文本编辑器，并执行以下操作：

   a. 注释掉`broker-to-broker-discovery-address`通过在行的开头放置哈希 (#) 来参数。
   b. 适用于`broker-to-broker-discovery-aws-region`中，输入您在其中创建应用程序负载均衡器的区域。
   c. 适用于`broker-to-broker-discovery-aws-alb-target-group-arn`中，输入与代理负载均衡器关联的目标组的 ARN。
   d. 保存并关闭 文件。

3. 停止代理实例。
4. 从已停止的代理实例创建 AMI。有关更多信息，请参阅 。从实例创建 Linux AMI中的适用于 Linux 实例的 Amazon EC2 用户指南.
5. 请使用 AMI 启动剩余的代理。
6. 将您创建的实例配置文件分配给所有 Broker 实例。
7. 将所有 Broker 实例注册为 Broker 负载均衡器的目标。有关更多信息，请参阅 。向您的目标组注册目标中的适用于应用程序负载均衡器的用户指南.

第 5 步：创建代理应用程序负载均衡器

创建应用程序负载均衡器以平衡代理和经纪商。

创建负载均衡器

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。

   在导航窗格中，选择负载均衡器然后选择创建 Load Balancer. 对于负载均衡器类型，请选择应用程序负载均衡器.
2. 适用于第 1 步：配置 Load Balancer中，执行以下操作：

   a. 适用于名称，输入一个描述性的负载均衡器名称。
   b. 适用于Scheme，选择面向 Internet 的.
   c. 适用于Load Balancer 协议，选择HTTPS，以及Load Balancer 端口输入，`8445`.
   d. 适用于VPC中，选择要使用的 VPC，然后选择该 VPC 中的所有子网。
   e. 选择下一步。
3. 适用于步骤 2: 配置安全设置中，执行以下操作：

   a. 适用于证书类型，选择从 ACM 中选择证书.
   b. 适用于证书名中，选择您之前请求的私有证书。
   c. 选择下一步。
4. 适用于步骤 3: 配置安全组，创建一个新的安全组，或者选择允许通过 HTTPS 和端口 8445 进行入站和出站流量的现有安全组。

   选择下一步。
5. 适用于步骤 4: 配置路由中，执行以下操作：

   a. 适用于目标组，选择新的目标组.
   b. 对于名称，输入目标组的名称。
   c. 适用于Target type，选择实例.
   d. 适用于协议，选择HTTPS. 对于端口，输入 `8445`。适用于协议版本，选择HTTP1.
   e. 对于运行状况检查协议，选择HTTPS，以及Path (路径)输入，`/health`.
   f. 选择下一步。
6. 适用于第 5 步：注册目标，选择所有经纪商实例并选择添加到已注册. 选择 Next:。审核。
7. 选择 Create (创建)。

第 6 步：启动代理

创建初始代理并将其配置为使用负载均衡器，从代理创建 AMI，然后使用 AMI 启动其余代理。这可确保所有代理都配置为使用相同的 Broker 证书和相同的负载均衡器配置。

启动代理

1. 准备 NICE DCV 服务器。有关更多信息，请参阅第 1 步：准备好好 DCV 服务器 (p. 5)。
2. 安装和配置代理。有关安装和配置 Agent 的更多信息，请参阅，请参阅第 3 步：设置 NICE DCV 会话管理器代理 (p. 7).

   Important

   修改 Agent 配置文件时，对于`broker_host`参数中，输入代理负载均衡器的 DNS。
3. 将 NICE DCV 服务器配置为使用 Broker 作为身份验证服务器。有关更多信息，请参阅第 4 步：将 NICE DCV 服务器配置为使用 Broker 作为身份验证服务器 (p. 10)。
4. 停止代理实例。
5. 从已停止的代理实例创建 AMI。有关更多信息，请参阅 。从实例创建 Linux AMI中的适用于 Linux 实例的 Amazon EC2 用户指南.

6. 使用 AMI 启动其余代理并将您创建的实例配置文件分配给所有代理。

使用标签来定位 NICE DCV 服务器

您可以为会话管理器代理分配自定义标签，以帮助识别和分类它们以及与之关联的 NICE DCV 服务器。创建新的 NICE DCV 会话时，您可以根据分配给各自代理的标签来定位一组 NICE DCV 服务器。有关如何基于代理标签定位 NICE DCV 服务器的更多信息，请参阅创建会话请求中的开发人员指南.

标签由标签密钥和值对组成，您可以使用对您的使用案例或环境有意义的任何信息对。您可以选择根据代理的主机的硬件配置标记代理。例如，您可以使用具有 4 GB 内存的主机标记所有 Agent`ram=4GB`. 或者你可以根据目的标记代理商。例如，您可以使用标记在生产主机上运行的所有 Agent`purpose=production`.

向代理分配标签

1. 例如，使用首选文本编辑器，创建一个新文件，然后为其指定一个描述性名称。`agent_tags.toml`. 文件类型必须是`.toml`，并且必须以 TOML 文件格式指定文件内容。
2. 在文件中，使用`key=value`格式的日期和时间。例如：

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

3. 打开 Agent 配置文件 (`/etc/dcv-session-manager-agent/agent.conf`Linux 或`C:\Program Files\NICE\DCVSessionManagerAgent\conf\agent.conf`适用于 Windows)。适用于`tags_folder`，然后指定标记文件所在目录的路径。

   如果目录包含多个标签文件，则跨文件定义的所有标签都将应用 Agent。这些文件以字母顺序读取。如果多个文件包含具有相同键的标签，则该值将被上次读取文件的值覆盖。
4. 保存并关闭 文件。
5. 停止代理，然后重新启动它。

   - Windows

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

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

   - Linux

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

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

配置外部授权服务器

授权服务器是负责对客户端 SDK 和 Agent 进行身份验证和授权的服务器。

默认情况下，会话管理器使用 Broker 作为授权服务器来为客户端 SDK 和 Agent 的软件语句生成 OAuth 2.0 访问令牌。如果您使用 Broker 作为授权服务器，则不需要额外的配置。

您可以将会话管理器配置为使用 Amazon Cognito 作为外部授权服务器而不是经纪商。有关 Amazon Cognito 的更多信息，请参阅Amazon Cognito 开发人员指南.

使用 Amazon Cognito 作为授权服务器

1. 创建新的 Amazon Cognito 用户池。有关用户池的更多信息，请参阅Amazon Cognito 的功能中的Amazon Cognito 开发人员指南.

   使用创建用户池命令，然后指定池名称和要在其中创建池名称的区域。

   在此示例中，我们将池命名为。dcv-session-manager-client-app然后我们在创建它us-east-1.

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

   输出示例

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

   记下userPoolId，您在下一个步骤中需要使用该值。
2. 为您的用户池创建新的域。使用创建用户池域命令，然后指定域名和userPoolId的用户池的值。

   在此示例中，域名为mydomain-544fa30f-c0e5-4a02-8d2a-a3761EXAMPLE然后我们在创建它us-east-1.

   ```
   $ 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
   ```

   输出示例

   ```
   {
       "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": {}
       }
   }
   ```

   用户池域的格式如下：https://*domain_name*.auth.*region*.amazoncognito.com. 在此示例中，用户池域为https://mydomain-544fa30f-c0e5-4a02-8d2a-a3761EXAMPLE.auth.us-east-1.amazoncognito.com.
3. 创建用户池客户端。使用创建用户池客户端命令并指定userPoolId所创建的用户池、客户端的名称以及要在其中创建它的区域。另外，请包括--generate-secret选项，以指定要为正在创建的用户池客户端生成密码的选项。

在这种情况下，客户端名为dcv-session-manager-client-app然后我们在us-east-1区域。

```
$ 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
```

输出示例

```
{
    "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

记下ClientId和ClientSecret. 当开发人员请求 API 请求访问令牌时，您需要向开发人员提供此信息。

4. 为用户池创建新的 OAuth2.0 资源服务器。资源服务器是访问受保护的资源的服务器。它处理已验证的访问令牌请求。

   使用创建资源服务器命令并指定userPoolId用户池、资源服务器的唯一标识符和名称、范围以及要在其中创建该服务器的区域。

   在此示例中，我们使用的是dcv-session-manager作为标识符和名称，我们使用sm_scope作为范围名称和描述。

   ```
   $ 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
   ```

   输出示例

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

5. 更新用户池客户端。

   使用更新用户池客户端命令。指定userPoolId在用户池中，ClientId的用户池客户端和区域。适用于--allowed-o-auth-flows，请指定client_credentials指示客户端应该使用客户端 ID 和客户端密钥的组合从令牌终端节点获取访问令牌。适用于--allowed-o-auth-scopes中，指定资源服务器标识符和范围名称，如下所示：*resource_server_identifier*/*scope_name*. 加入--

`allowed-o-auth-flows-user-pool-client`表示在与 Cognito 用户池进行交互时允许客户端遵循 OAuth 协议。

```
$ 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
```

输出示例

```
{
    "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

用户池现在已准备好提供和验证访问令牌。在此示例中，授权服务器的 URL 为`https://cognito-idp.us-east-1.amazonaws.com/us-east-1_QLEXAMPLE/.well-known/jwks.json`.

6. 测试配置。

```
$ 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"
```

输出示例

```
{
"access_token":"eyJraWQiOiJGQ0VaRFpJUUptT3NSaW41MmtqaDdEbTZYb0RnSTQ5b2VUT0cxUUI1Q2VJPSIsImFsZyI6IlJ
Zkfi0HIDsd6audjTXKzHlZGScr6ROdZtId5dThkpEZiSx0YwiiWe9crAlqoazlDcCsUJHIXDtgKW64pSj3-
uQQGg1Jv_tyVjhrA4JbD0k67WS2V9NW-
uZ7t4zwwaUmOi3KzpBMi54fpVgPaewiVlUm_aS4LUFcWT6hVJjiZF7om7984qb2gOa14iZxpXPBJTZX_gtG9EtvnS9uW0QygTJR
"expires_in":3600,
"token_type":"Bearer"
}
```

7. 注册外部授权服务器以便在经纪商中使用注册身份验证服务器 (p. 30)命令。

```
$ 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
```

开发人员现在可以使用服务器请求访问令牌。请求访问令牌时，请提供在此生成的客户端 ID、客户端密钥和服务器 URL。有关请求访问令牌的更多信息，请参阅创建获取访问令牌并发出 API 请求中的NICE DCV 会话管理器开发者指南.

配置代理持久性

Session Manager 经纪人支持与外部数据库的集成。外部数据库允许 Session Manager 保留状态数据和密钥，以便之后可用。事实上，代理数据分布在集群上，如果主机需要重新启动或集群终止，则很容易丢失数据。启用此功能后，您可以添加和删除经纪商节点。此外，您可以停止集群并重新启动它，而无需重新生成密钥或丢失关于哪个 NICE DCV Server 处于打开或关闭状态的信息。

以下类型的信息可设置为持久保留：

- 用于设置会话以建立与客户端连接的密钥
- 在行中会话数据
- NICE DCV 服务器状态

NICE DCV 会话管理器支持 DynamoDB、MariaDB 和 MySQL 数据库。您必须设置和管理其中一个数据库才能使用此功能。如果您的经纪机器托管在 Amazon EC2 上，我们建议使用 DynamoDB 作为外部数据库，因为它不需要任何额外的设置。

Note

运行外部数据库时，可能会产生额外费用。要查看有关 DynamoDB 定价的信息，请参阅预置容量的定价.

将经纪商配置为持续在 DynamoDB 上

配置经纪商以开始在 DynamoDB 上存储数据：

1. 打开`/etc/dcv-session-manager-broker/session-manager-broker.properties`使用您常用的文本编辑器并进行以下编辑：
   - Set `enable-persistence = true`
   - Set `persistence-db = dynamodb`
   - 适用于`dynamodb-region`指定要存储包含代理数据的表的 &aws; 区域。有关支持的区域列表，请参阅DynamoDB 服务终端节点.
   - 适用于`dynamodb-table-rcu`指定每个表支持的读取容量单位 (RCU) 的数量。有关 RCU 的更多信息，请参阅DynamoDB 预置容量.
   - 适用于`dynamodb-table-wcu`指定每个表支持的写入容量单位 (WCU) 的数量。有关 WCU 的更多信息，请参阅DynamoDB 预置容量.
   - 对于 dynamodb-table 名称前缀，请指定添加到每个 DynamoDB 表的前缀（有助于区分使用同一账户的多个经纪商集群）。只允许使用字母数字字符、点、短划线和下划线。
2. 停止集群中的所有代理。对于每个代理，运行以下命令：

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

3. 确保集群中的所有经纪商都已停止，然后重新启动所有经纪商。通过运行以下命令启动每个代理：

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

代理主持人必须拥有调用 DynamoDB API 的权限。在 Amazon EC2 实例上，使用 Amazon EC2 元数据服务自动检索凭证。如果需要指定不同的凭据，可以使用受支持的凭据检索技术之一（例如 Java 系统属性或环境变量）对其进行设置。有关更多信息，请参阅 。提供和检索 &aws; 凭证.

将经纪商配置为在 MariaDB/MySQL 上持续存在

Note

这些区域有：`/etc/dcv-session-manager-broker/session-manager-broker.properties`文件包含敏感数据。默认情况下，其写入访问权限限制为 root 用户，其读取访问权限仅限于 root 用户和运行 Broker 的用户。默认情况下为`dcvsmbroker`用户。Broker 在启动时检查文件是否具有预期的权限。

配置经纪商以开始在 MariaDB/MySQL 上保留他们的数据：

1. 打开`/etc/dcv-session-manager-broker/session-manager-broker.properties`使用您常用的文本编辑器进行以下编辑：
   - Set `enable-persistence = true`
   - Set `persistence-db = mysql`
   - Set `jdbc-connection-url = jdbc:mysql://<db_endpoint>:<db_port>/<db_name>?createDatabaseIfNotExist=true`

     在此配置中，<db_endpoint>是数据库终端节点，<db_port>是数据库端口，<db_name>是数据库名称。
   - 适用于`jdbc-user`指定有权访问数据库的用户的名称。
   - 适用于`jdbc-password`指定有权访问数据库的用户的密码。
2. 停止集群中的所有代理。对于每个代理，运行以下命令：

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

3. 确保集群中的所有经纪商都已停止，然后重新启动所有经纪商。对于每个代理，运行以下命令：

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

与 NICE DCV 连接网关集成

NICE DCV 连接网关是一个可安装的软件包，使用户能够通过 LAN 或 VPC 的单个接入点访问一组 NICE DCV 服务器。

如果您的基础设施包括可通过 NICE DCV 连接网关访问的 NICE DCV 服务器，则可以配置会话管理器以集成 NICE DCV 连接网关。按照以下部分中概述的步骤，经纪商将充当会话解析程序对于连接网关。换而言之，代理将公开额外的 HTTP 终端节点。连接网关将对端点进行 API 调用，以检索将 NICE DCV 连接路由到代理选择的主机所需的信息。

将会话管理器 Broker 设置为 NICE DCV 连接网关的会话解析器

代理会话管理器

1. 打开`/etc/dcv-session-manager-broker/session-manager-broker.properties`使用首选文本编辑器并应用以下更改：

- Set `enable-gateway = true`
- Set`gateway-to-broker-connector-https-port`到免费的 TCP 端口（默认值为 8447）
- Set`gateway-to-broker-connector-bind-host`转换为代理绑定 NICE DCV 连接网关连接的主机的 IP 地址（默认值为 0.0.0.0）

2. 然后运行以下命令以停止并重启代理：

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

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

3. 检索经纪商的自签名证书的副本并将其放在用户目录中。

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

下一步中安装 NICE DCV 连接网关时需要使用该值。

NICE DCV 连接网关侧

- 请按照部分在 NICE DCV 连接网关文档中。

  由于 NICE DCV 连接网关向代理发出 HTTP API 调用，因此如果代理使用的是自签名证书，则需要将代理证书复制到 NICE DCV 连接网关主机（在上一步中检索到）并设置`ca-file`中的参数`[resolver]`NICE DCV 连接网关配置的部分。

可选-启用 TLS 客户端验证

完成上一步后，会话管理器和连接网关就可以通过安全通道进行通信，连接网关可以在其中验证会话管理器经纪人的身份。如果您还要求 Session Manager Broker 在建立安全通道之前验证连接网关的身份，则需要按照下一节中的步骤启用 TLS 客户端身份验证功能。

> **Note**
>
> 如果会话管理器位于负载均衡器后面，则无法使用具有 TLS 连接终止的负载均衡器（例如应用程序负载均衡器 (ALB) 或网关负载均衡器 (GLB)）启用 TLS 客户端身份验证。只能支持没有终止 TLS 的负载均衡器，例如网络负载均衡器 (NLB)。如果您使用 ALB 或 GLB，则可以强制只有特定的安全组可以联系负载均衡器，从而确保额外的安全级别；有关安全组的更多信息请点击此处：您的 VPC 的安全组

代理会话管理器

1. 要为会话管理器经纪商和 NICE DCV 连接网关之间的通信启用 TLS 客户端身份验证，请按照以下步骤操作：
2. 运行以下命令生成所需的密钥和证书：命令的输出将告诉你生成凭据的文件夹以及用于创建 TrustStore 文件。

```
sudo /usr/share/dcv-session-manager-broker/bin/dcv-session-manager-broker/gateway-creds-generation.sh
```

3. 将 NICE DCV 连接网关的私钥和自签名证书的副本放在用户目录中。在下一步中，当你在 NICE DCV 连接网关中启用 TLS 客户端身份验证时，你将需要它。

```
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. 然后使用首选的文本编辑器打开 /etc/dcv-会议经理器/会议管理器/Broker .properties，然后执行以下操作：
   - Setenable-tls-client-auth-gateway到true
   - Setgateway-to-broker-connector-trust-store-file走向 TrustStore 在上一步中创建的文件
   - Setgateway-to-broker-connector-trust-store-pass转到用于创建 TrustStore 上一步中的文件
5. 然后运行以下命令以停止并重启代理：

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

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

NICE DCV 连接网关侧

- 请按照部分在 NICE DCV 连接网关文档中。
  - 设置时，使用您在上一步中复制的证书文件的完整路径cert-file中的参数[resolver]部分
  - 设置时，使用您在上一步中复制的密钥文件的完整路径cert-key-file中的参数[resolver]部分

NICE DCV 会话管理器 NICE DCV 服务器-DNS 映射

NICE DCV 连接网关需要 NICE DCV 服务器的 DNS 名称才能连接到 DCV 服务器实例。本节说明如何定义 JSON 文件，其中包含每个 DCV 服务器及其关联的 DNS 名称之间的映射。

文件结构

映射由具有以下字段的 JSON 对象列表组成：

```
[
 {
  “ServerIdType”: "Ip",
  “ServerId”: "192.168.0.1",
  “DnsNames”:
  {
   “InternalDnsName”: "internal"
  }
 },
 ...
]
```

其中：

`ServerIdType:`

标识值所指的 ID 类型；目前可用的值为 IPAddress、Agent ServerID 和 instanceId：

`Ip:`

同时适用于 Amazon EC2 和本地基础设施；系统管理员可以使用 ifconfig (Linux) 或 ipconfig (Windows) 命令快速检索。在中也提供了此信息 DescribeServers API 响应。

`Id:`

同时适用于 Amazon EC2 和本地基础设施；每次主机名或 IP 地址更改时，会话管理器代理都会创建一个新的 UUID。此信息可在 DescribeServers API 响应。

`Host.Aws.Ec2InstanceId:`

它仅适用于 Amazon EC2 实例，唯一标识计算机；实例重启后不会更改。可以通过联系 http://169.254.169.254/latest/meta-data/instance-id 在房东上检索。在中也提供了此信息 DescribeServers API 响应。

`ServerId:`

指定类型的 ID，用于唯一标识网络中的每个 NICE DCV 服务器。

`DnsNames:`

包含与 NICE DCV 服务器关联的 DNS 名称的对象此对象将包含：

`InternalDnsNames:`

NICE DCV 连接网关用于连接到实例的 DNS 名称。

请使用会话管理器 Broker CLI 命令`register-server-dns-mapping`要从文件加载映射（命令页参考：注册服务器-dns 映射）和`describe-server-dns-mappings`列出当前在会话管理器 Broker 中加载的映射（命令页参考：描述服务器-dns-映射）。

持久性

我们强烈建议您启用 Session Manager Broker 的持久性功能，以防止在多个经纪商或整个集群关闭时出现映射丢失。有关启用数据持久性的更多信息，请参阅配置代理持久性

与 Amazon CloudWatch 集成

会话管理器支持与 Amazon CloudWatch 针对在 Amazon EC2 实例上运行的经纪商以及在本地主机上运行的经纪商集成。

Amazon CloudWatch 可实时监控您的亚马逊云科技 (Amazon) 资源以及您在 Amazon 上运行的应用程序。您可以使用 CloudWatch 收集和跟踪指标，这些指标是您可衡量的相关资源和应用程序的变量。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

您可以将会话管理器代理配置为向 Amazon CloudWatch 发送以下指标数据：

- `Number of DCV servers`— 经纪商管理的 DCV 服务器数量。
- `Number of ready DCV servers`— 中的 DCV 服务器数量`READY`由经纪商管理的州。
- `Number of DCV sessions`— 经纪商管理的 DCV 会话数量。
- `Number of DCV console sessions`— 由经纪商管理的 DCV 控制台会话数。
- `Number of DCV virtual sessions`— 经纪商管理的 DCV 虚拟会话数量。
- `Heap memory used`— Broker 使用的堆内存量。
- `Off-heap memory used`— 代理使用的堆外内存量。
- `Describe sessions request time`— 完成 DescribeSessions API 请求所花的时间量。
- `Delete sessions request time`— 完成删除会话 API 请求所花的时间量。
- `Create sessions request time`— 完成 CreateSessions API 请求所花的时间量。
- `Get session connection data request time`— 完成 getSessionConnection Data API 请求所花的时间量。
- `Update session permissions sequest time`— 完成更新会话权限 API 请求所花的时间量。

将经纪商配置为向 Amazon CloudWatch 发送指标数据

1. 打开`/etc/dcv-session-manager-broker/session-manager-broker.properties`使用首选文本编辑器，然后执行以下操作：
   - Se`enable-cloud-watch-metrics`到`true`
   - 适用于`cloud-watch-region`中，指定要在其中收集指标数据的区域。

     Note

     如果您的经纪商在 Amazon EC2 实例上运行，则此参数为可选参数。区域将自动从实例元数据服务 (IMDS) 中检索。如果您在本地主机上运行 Broker，则此参数是必需的。
2. 停止并重新启动经纪商。

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

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

经纪商主持人还必须有权调用`cloudwatch:PutMetricData`API。Amazon可以使用受支持的凭据检索技术之一检索凭据。有关更多信息，请参阅 。提供和检索Amazon凭证.

升级 NICE DCV 会话管理器

以下主题介绍如何升级会话管理器。

Note

我们强烈建议您在升级会话管理器经纪人之前升级所有会话管理器代理，以避免在引入新功能时出现不兼容问题。

主题

升级 NICE DCV 会话管理器代理

Linux host

Note

以下说明用于在 64 位 x86 主机上安装代理。要在 64 位 ARM 主机上安装代理，对于亚马逊 Linux、RHEL 和 Centos，请替换*x86_64*和aarch64，对于 Ubuntu，替换*amd64*和arm64.

在 Linux 主机上更新代理

1. 运行以下命令以停止代理。

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

2. 下载安装包。

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

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/nice-dcv-session-manager-agent-2022.0.520-1.el7.x86_64.rpm
```

   - RHEL 8.x 和 CentOS 8.x

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/nice-dcv-session-manager-agent-2022.0.520-1.el8.x86_64.rpm
```

   - Ubuntu 18.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/nice-dcv-session-manager-agent_2022.0.520-1_amd64.ubuntu1804.deb
```

   - Ubuntu 20.04

```
$  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/nice-dcv-session-manager-agent_2022.0.520-1_amd64.ubuntu2004.deb
```

   - SUSE Linux Enterprise 12

```
$  curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/
nice-dcv-session-manager-agent-2022.0.520-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$  curl -O https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerAgents/
nice-dcv-session-manager-agent-2022.0.520-1.sles15.x86_64.rpm
```

3. 安装 软件包。

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

```
$  sudo yum install -y nice-dcv-session-manager-agent-2022.0.520-1.el7.x86_64.rpm
```

- RHEL 8.x 和 CentOS 8.x

```
$  sudo yum install -y nice-dcv-session-manager-agent-2022.0.520-1.el8.x86_64.rpm
```

- Ubuntu 18.04

```
$  sudo apt install ./nice-dcv-session-manager-
agent_2022.0.520-1_amd64.ubuntu1804.deb
```

- Ubuntu 20.04

```
$  sudo apt install ./nice-dcv-session-manager-
agent_2022.0.520-1_amd64.ubuntu2004.deb
```

- SUSE Linux Enterprise 12

```
$  sudo zypper install nice-dcv-session-manager-
agent-2022.0.520-1.sles12.x86_64.rpm
```

- SUSE Linux Enterprise 15

```
$  sudo zypper install nice-dcv-session-manager-
agent-2022.0.520-1.sles15.x86_64.rpm
```

4. 运行以下命令以启动代理。

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

Windows host

在 Windows 主机上更新代理

1. 停止代理服务。在命令提示符下运行以下命令。

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

2. 下载安装代理.
3. 运行安装程序。在欢迎屏幕上，选择 Next。
4. 在 EULA 屏幕上，仔细阅读许可协议，如果您同意，请选择我接受这些条款然后选择下一步.
5. 要开始安装，请选择安装.
6. 重新启动代理服务。在命令提示符下运行以下命令。

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

升级 NICE DCV 会话管理器经纪商

升级经纪商

1. Connect 到您打算升级经纪商的主机。
2. 停止经纪服务。

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

3. 下载安装包。

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

   ```
   $  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerBrokers/nice-dcv-session-manager-broker-2022.0.341-1.el7.noarch.rpm
   ```

   - RHEL 8.x 和 CentOS 8.x

   ```
   $  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerBrokers/nice-dcv-session-manager-broker-2022.0.341-1.el8.noarch.rpm
   ```

   - Ubuntu 18.04

   ```
   $  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerBrokers/nice-dcv-session-manager-broker-2022.0.341-1_all.ubuntu1804.deb
   ```

   - Ubuntu 20.04

   ```
   $  wget https://d1uj6qtbmh3dt5.cloudfront.net/2022.0/SessionManagerBrokers/nice-dcv-session-manager-broker-2022.0.341-1_all.ubuntu2004.deb
   ```

4. 安装 软件包。

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

   ```
   $  sudo yum install -y nice-dcv-session-manager-broker-2022.0.341-1.el7.noarch.rpm
   ```

   - RHEL 8.x 和 CentOS 8.x

   ```
   $  sudo yum install -y nice-dcv-session-manager-broker-2022.0.341-1.el8.noarch.rpm
   ```

   - Ubuntu 18.04

   ```
   $  sudo apt install -y nice-dcv-session-manager-
   broker-2022.0.341-1_all.ubuntu1804.deb
   ```

   - Ubuntu 20.04

   ```
   $  sudo apt install -y nice-dcv-session-manager-
   broker-2022.0.341-1_all.ubuntu2004.deb
   ```

5. 启动 Broker 服务并确保它在每次实例启动时自动启动。

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

经纪商 CLI 引用

本部分介绍如何使用 Broker 命令行界面 (CLI) 命令。

如果使用外部身份验证服务器生成 OAuth 2.0 访问令牌，请使用以下命令：

- 注册身份验证服务器 (p. 30)
- 列表身份验证服务器 (p. 31)
- 取消注册身份验证服务器 (p. 32)

如果您使用会话管理器 Broker 作为 OAuth 2.0 身份验证服务器，请使用以下命令。

- 注册 api-Client (p. 33)
- 描述 api 客户端 (p. 34)
- 取消注册 api-Client (p. 35)
- 续订身份验证服务器 api-key (p. 35)

使用以下命令管理会话管理器代理。

- 生成软件声明 (p. 36)
- 描述软件声明 (p. 37)
- 停用软件声明 (p. 38)
- 描述代理-客户端 (p. 39)
- 取消注册代理-客户端 (p. 40)

使用以下命令管理 DCV 服务器-DNS 名称映射文件。

- 注册服务器-dns-映射 (p. 40)
- 描述服务器-dns-映射 (p. 41)

注册身份验证服务器

注册外部身份验证服务器以便与经纪商一起使用。

默认情况下，会话管理器使用 Broker 作为身份验证服务器来生成 OAuth 2.0 访问令牌。如果将 Broker 用作身份验证服务器，则不需要额外的配置。

但是，如果您选择使用外部身份验证服务器，例如 Active Directory 或 Amazon Cognito，则必须使用此命令注册外部身份验证服务器。

主题

- 语法 (p. 31)
- 选项 (p. 31)
- 示例 (p. 31)

语法

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

选项

--url

要使用的外部身份验证服务器的 URL。你必须追加.well-known/jwks.json转到身份验证服务器 URL。

类型: 字符串

：必需 是

示例

以下示例通过 URL 注册外部身份验证服务器：https://my-auth-server.com/.

命令

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

输出

```
Jwk url registered.
```

列表身份验证服务器

列出已注册的外部身份验证服务器。

主题

语法

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

输出

Urls

已注册的外部身份验证服务器的 URL。

示例

以下示例列出了所有已注册的外部身份验证服务器。

命令

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

输出

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

取消注册身份验证服务器

取消注册外部身份验证服务器。取消注册外部身份验证服务器后，它将不能再用于生成 OAuth 2.0 访问令牌。

主题

语法

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

选项

`--url`

要取消注册的外部身份验证服务器的 URL。你必须追加`.well-known/jwks.json`转到身份验证服务器 URL。

类型: 字符串

：必需 是

输出

`Url`

未注册的外部身份验证服务器的 URL。

示例

以下示例通过 URL 注册外部身份验证服务器：`https://my-auth-server.com/`.

命令

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

输出

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

注册 api-Client

向 Broker 注册会话管理器客户端并生成客户端凭证，客户端可以使用这些凭据来检索 OAuth 2.0 访问令牌，这是发出 API 请求所需的。

> Important
>
> 确保将凭证存储在安全位置。它们以后都无法恢复。

仅当经纪人用作 OAuth 2.0 身份验证服务器时，才使用此命令。

主题

语法

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

选项

--name

用于标识会话管理器客户端的唯一名称。

类型: 字符串

：必需 是

输出

client-id

会话管理器客户端用于检索 OAuth 2.0 访问令牌的唯一客户端 ID。

client-password

会话管理器客户端用于检索 OAuth 2.0 访问令牌的密码。

示例

以下示例注册名为：my-sm-client.

命令

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

输出

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

描述 api 客户端

列出已在经纪商注册的会话管理器客户端。

主题

语法

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

输出

name

会话管理器客户端的唯一名称。

id

会话管理器客户端的唯一 ID。

active

指示会话管理器客户端的状态。如果客户端处于活动状态，则值为true；否则为false.

示例

以下示例列出了注册的会话管理器客户端。

命令

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

输出

```
Api clients
```

```
[ {
"name" : "client-abc",
"id" : "f855b54b-40d4-4769-b792-b727bEXAMPLE",
"active" : false
}, {
"name" : "client-xyz",
"id" : "21cfe9cf-61d7-4c53-b1b6-cf248EXAMPLE",
"active" : true
}]
```

取消注册 api-Client

停用注册的会话管理器客户端。已停用的会话管理器客户端不能再使用其凭据来检索 OAuth 2.0 访问令牌。

主题

语法

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

选项

--client -id

要停用的会话管理器客户端的 ID。

类型: 字符串

：必需 是

示例

以下示例禁用客户端 ID 为的会话管理器客户端`f855b54b-40d4-4769-b792-b727bEXAMPLE`.

命令

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

输出

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

续订身份验证服务器 api-key

续订经纪商用于签署出售给会话管理器客户端的 OAuth 2.0 访问令牌的公钥和私钥。如果您续订密钥，则必须向开发人员提供新的私钥，因为它是发出 API 请求所需的。

主题

语法

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

示例

以下示例更新了公有密钥和私有密钥。

命令

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

输出

```
Keys renewed.
```

生成软件声明

生成软件声明。

代理商必须在经纪商注册才能启用通信。代理商需要软件声明才能向经纪商注册。在代理获得软件声明后，它可以使用OAuth 2.0 动态客户端注册协议. 代理在经纪商注册后，它会收到一个客户 ID 和客户密码，用于向经纪商进行身份验证。

Broker 和 Agent 在首次安装时会收到并使用默认的软件语句。您可以继续使用默认软件语句，也可以选择生成新的软件语句。如果生成新的软件语句，则必须将软件语句放入 Agent 上的新文件中，然后将文件路径添加到`agent.software_statement_path`中的参数`agent.conf`文件。完成此操作后，请停止并重新启动代理，以便它可以使用新的软件声明向经纪商注册。

主题

语法

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

输出

`software-statement`

软件陈述。

示例

以下示例生成软件语句。

命令

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

输出

```
software-statement:
 ewogICJpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiYWN0aXZlIiA6IHRydWUsCiAgImlzc3
```

描述软件声明

描述现有的软件语句。

主题

- 语法 (p. 31)
- 输出 (p. 31)
- 示例 (p. 31)

语法

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

输出

software-statement

软件陈述。

issued-at

软件的生成日期和时间。

is-active

软件语句的当前状态。`true`软件语句是否处于活动状态；否则是`false`.

示例

以下示例生成软件语句。

命令

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

输出

```
Software Statements
[ {
"software-statement" :
 "ewogICJpZCIgOiAiYmEEXAMPLEYtNzUwNy00YmFhLTliZWItYTA1MmJjZTE3NDJjIiwKICAiaXNzdWVkQXQiIDogMTU5NjY0MTkxM
"issued-at" : "2020.08.05 15:38:32 +0000",
"is-active" : "true"
}, {
"software-statement" :
 "EXAMPLEpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiaXNzdWEXAMPLEDogMTU5Njc5NTg4M
"issued-at" : "2020.08.07 10:24:41 +0000",
"is-active" : "true"
} ]
```

停用软件声明

停用软件语句。当您停用软件语句时，它不能再用于代理注册。

主题

语法

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

选项

--software-statement

要停用的软件声明。

类型: 字符串

：必需 是

示例

以下示例禁用软件语句。

命令

```
sudo -u root dcv-session-manager-broker deactivate-software-statement --software-statement
 EXAMPLEpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiaXNEXAMPLEQiIDogMTU5Njc5NTg4MS
```

输出

```
Software statement
 EXAMPLEpZCIgOiAiYjc1NTVhN2QtNWI0MC00OTJhLWJjOTUtNmUzOWNhYzkxMDcxIiwKICAiaXNEXAMPLEQiIDogMTU5Njc5NTg4MS
 deactivated
```

描述代理-客户端

描述在经纪商注册的代理商。

主题

语法

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

输出

name

代理的名称。

id

代理的唯一 ID。

active

代理的状态。`true`代理是否处于活动状态；否则是`false`.

示例

以下示例描述了代理。

命令

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

输出

```
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
```

```
} ]
```

取消注册代理-客户端

从经纪商注销代理商。

主题

语法

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

选项

--client-id

要取消注册的代理的 ID。

类型: 字符串

: 必需 是

示例

以下示例取消注册 Agent。

命令

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

输出

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

注册服务器-dns-映射

注册 DCV 服务器-来自 JSON 文件的 DNS 名称映射。

语法

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

选项

--file-path

文件的路径，文件中包含 DCV 服务器-DNS 名称映射。

类型: 字符串

：必需 是

示例

以下示例注册了 DCV 服务器-来自文件 /tmp/mapings.json 的 DNS 名称映射。

命令

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

输出

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

描述服务器-dns-映射

描述当前可用的 DCV 服务器-DNS 名称映射。

语法

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

输出

serverIdType

服务器 ID 的类型。

serverId

服务器的唯一 ID。

dnsNames

内部和外部 DNS 名称

internalDnsNames

内部 DNS 名称

externalDnsNames

外部 DNS 名称

示例

以下示例列出了注册的 DCV 服务器-DNS 名称映射。

命令

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

输出

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

配置文件引用

本节提供有关代理和代理配置文件的信息。

主题

代理配置文件

经纪商配置文件 (`/etc/dcv-session-manager-broker/session-manager-broker.properties`) 包括可配置用于自定义会话管理器功能的参数。您可以使用首选文本编辑器编辑配置文件。

> **Note**
>
> 这些区域有：`/etc/dcv-session-manager-broker/session-manager-broker.properties`文件包含敏感数据。默认情况下，其写入访问权限限制为 root 用户，其读取访问权限仅限于 root 用户和运行 Broker 的用户。默认为`dcvsmbroker`用户。经纪商在启动时检查文件是否具有预期的权限。

下表列出了代理配置文件中的参数。

参数名称	必填	默认值	说明
`broker-java-home`	否		指定 Broker 将使用的 Java 主目录的路径，而不是系统默认目录的路径。如果设置，经纪商将使用`<broker-java-home>/bin/java`启动时。 提示：Broker 需要 Java 运行时环境 11，如果作为成功安装的依赖关系丢失，它将被安装。如果版本 11 未设置为默认 Java 环境，则可以使用以下命令抓取其主目录： `$ sudo alternatives --display java`
`session-screenshot-max-width`	否	`160`	使用获取会话截图API。
`session-screenshot-max-height`	否	`100`	使用获取会话截图API。
`session-screenshot-format`	否	`png`	使用获取会话截图API。

参数名称	必填	默认值	说明
create-sessions-queue-max-size	否	1000	未配送的最大数目创建会话可加入队列的 API 请求。队列已满时，新的未履行的请求将被拒绝。
create-sessions-queue-max-time-seconds	否	1800	未完成的最长时间（以秒为单位）创建会话API 请求可以保留在队列中。如果无法在指定的时间内完成请求，则会失败。
session-manager-working-path	是	/tmp	指定 Broker 在其中写入操作所需文件的目录的路径。此目录必须只能由经纪商访问。
enable-authorization-server	是	true	指定代理是否是用于为客户端 API 生成 OAuth 2.0 访问令牌的身份验证服务器。
enable-authorization	是	true	启用或禁用客户端授权。如果启用客户端授权，则在发出 API 请求时，客户端 API 必须提供访问令牌。如果禁用客户端授权，客户端 API 可以发出没有访问令牌的请求。
enable-agent-authorization	是	true	启用或禁用代理授权。如果启用代理授权，则代理在与经纪商通信时必须提供访问令牌。
delete-session-duration-hours	否	1	指定已删除的会话变为不可见且不再返回的小时数DescribeSessionAPI 调用。
connect-session-token-duration-minutes	否	60	指定分钟数 ConnectSession 令牌仍然有效。
client-to-broker-connector-https-port	是		指定 Broker 在其中监听客户端连接的 HTTPS 端口。

参数名称	必填	默认值	说明
client-to-broker-connector-bind-host	否	0.0.0.0	指定 Broker 绑定客户端连接的主机的 IP 地址。
client-to-broker-connector-key-store-file	是		指定用于 TLS 客户端连接的密钥存储库。
client-to-broker-connector-key-store-pass	是		指定密钥存储通行证。
agent-to-broker-connector-https-port	是		指定代理监听代理连接的 HTTPS 端口。
agent-to-broker-connector-bind-host	否	0.0.0.0	指定代理绑定用于代理连接的主机的 IP 地址。
agent-to-broker-connector-key-store-file	是		指定用于 TLS 代理连接的密钥存储库。
agent-to-broker-connector-key-store-pass	是		指定密钥存储通行证。

参数名称	必填	默认值	说明
broker-to-broker-port	是		指定用于经纪商到经纪商连接的端口。
broker-to-broker-bind-host	否	0.0.0.0	指定经纪商绑定的主机的 IP 地址以获得经纪商到经纪商的连接。
broker-to-broker-discovery-port	是		指定经纪商用来互相发现的端口。
broker-to-broker-discovery-addresses	否		指定在*ip_address*：*#*#格式的日期和时间。如果有多个经纪商，请使用逗号分隔值。如果你指定broker-to-broker-discovery-multicast-group、broker-to-broker-discovery-multicast-port、broker-to-broker-discovery-Amazon-region，或者broker-to-broker-discovery-Amazon-alb-target-group-arn中，省略此参数。
broker-to-broker-discovery-multicast-group	否		指定用于经纪商对手发现的多播组。如果你指定broker-to-broker-discovery-addresses、broker-to-broker-discovery-aws-region，或者broker-to-broker-discovery-Amazon-alb-target-group-arn中，省略此参数。

参数名称	必填	默认值	说明
broker-to-broker-discovery-multicast-port	否		指定用于 Broker-Ooker 发现的组播端口。如果你指定broker-to-broker-discovery-addresses、broker-to-broker-discovery-Amazon-region，或者broker-to-broker-discovery-Amazon-alb-target-group-arn中，省略此参数。
broker-to-broker-discovery-Amazon-region	否		指定Amazon用于代理到代理发现的应用程序负载均衡器的区域。如果你指定broker-to-broker-discovery-multicast-group、broker-to-broker-discovery-multicast-port，或者broker-to-broker-discovery-addresses中，省略此参数。
broker-to-broker-discovery-Amazon-alb-target-group-arn	否		用于经纪商到经纪商发现的应用程序负载均衡器目标组用户的 ARN。如果你指定broker-to-broker-discovery-multicast-group、broker-to-broker-discovery-multicast-port，或者broker-to-broker-discovery-addresses中，省略此参数。
broker-to-broker-distributed-memory-max-size-mb	否	4096	指定单个 Broker 用于存储 NICE DCV 会话数据的最大堆外内存量。
broker-to-broker-key-store-file	是		指定用于 TLS Broker 连接的密钥存储库。

参数名称	必填	默认值	说明
broker-to-broker-key-store-pass	是		指定密钥存储通行证。
enable-cloud-watch-metrics	否	false	启用或禁用亚马逊 CloudWatch 指标。如果启用 CloudWatch 指标，您可能需要为cloud-watch-region.
cloud-watch-region	否	仅当为时必需enable-cloud-watch-metrics设置为true. 如果代理安装在 Amazon EC2 实例上，则从 IMDS 中检索该区域。	这些区域有：Amazon在哪个地区 CloudWatch 指标已发布。
每秒最大的 api 请求	否	1000	指定 Broker API 在限制之前每秒可以处理的最大请求数。
enable-throttling-forwarded-for-header	否	false	如果设置为true限制将从 X-Forwared-For 标头中检索来电者 IP（如果存在）。
create-sessions-number-of-retries-on-failure	否	2	指定在 NICE DCV 服务器主机上创建会话请求失败后要执行的最大重试次数。设置为 0 可在失败时永远不执行重试。
autorun-file-arguments-max-size	否	50	指定可传递给自动运行文件的参数的最大数目。
autorun-file-arguments-max-argument-length	否	150	指定每个自动运行文件参数的最大长度（以字符为单位）。
enable-persistence	是	false	如果设置为true，经纪人状态数据保存在外部数据库中。
persistence-db	否	仅当为时必需enable-persistence设置为true.	指定用于持久性的数据库。唯一支持的值为：dynamodb和mysql.

参数名称	必填	默认值	说明
dynamodb-region	否	仅当为时必需enable-persistence设置为true和persistence-db设置为dynamodb.	指定创建和访问 DynamoDB 表的区域。
dynamodb-table-rcu	否	仅当为时必需enable-persistence设置为true和persistence-db设置为dynamodb.	指定每个 DynamoDB 表的读取容量单位 (RCU) 有关 RCU 的详细信息，请参阅预置容量的定价.
dynamodb-table-wcu	否	仅当为时必需enable-persistence设置为true和persistence-db设置为dynamodb.	指定每个 DynamoDB 表的写入容量单位 (WCU)。有关 WCU 的更多信息，请参阅预置容量的定价.
dynamodb-table-name-prefix	否	仅当为时必需enable-persistence设置为true和persistence-db设置为dynamodb.	指定添加到每个 DynamoDB 表的前缀（有助于使用相同的代理集群区分多个代理集群）Amazon账户)。只允许使用字母数字字符、点、短划线和下划线。
jdbc-connection-url	否	仅当为时必需enable-persistence设置为true和persistence-db设置为mysql.	指定到 MariaDB/MySQL 数据库的连接 URL；它包含终端节点和数据库名称。url 应该有这种格式： `jdbc:mysql://<db_endpoint>:<db_port>/<db_name>?createDatabaseIfNotExist=true` 其中<db_endpoint>是 MariaDB/MySQL 数据库终端节点，<db_port>是数据库端口<db_name>是数据库名称。
jdbc-user	否	仅当为时必需enable-persistence设置为true和persistence-db设置为mysql.	指定有权访问 MariaDB/MySQL 数据库的用户的名称。
jdbc-password	否	仅当为时必需enable-persistence设置为true和persistence-db设置为mysql.	指定有权访问 MariaDB/MySQL 数据库的用户的密码。
seconds-before-deleting-unreachable-dcv-server	否	1800	指定从系统中删除无法访问的服务器的秒数。

代理配置文件

代理配置文件 (/etc/dcv-session-manager-agent/agent.confLinux 和C:\Program Files\NICE\DCVSessionManagerAgent\conf\agent.confWindows) 包含可配置用于自定义会话管理器功能的参数。您可以使用首选文本编辑器编辑配置文件。

下表列出了代理配置文件中的参数。

参数名称	必填	默认值	说明
agent.broker_host	是		指定代理主机的 DNS 名称。
agent.broker_port	是		指定与经纪商进行通信的端口。
agent.ca_file	否		只有在需要tls_strict设置为true. 指定验证 TLS 证书所需的证书 (.pem) 文件的路径。将自签名证书从代理复制到代理。
agent.init_folder	否	• /var/lib/dcv-session-manager-agent/init(Linux)	指定主机服务器上用于存储创建 NICE DCV 服务器会话时允许初始化自定义脚本的文件夹的路径。您必须指定绝对路径。必须可以访问该文件夹并且文件必须可由使用initFile的请求参数创建会话API。
agent.tls_strict	否	true	指示是否应使用严格的 TLS 验证。
agent.software_statement_path	否		仅当未使用默认软件语句时才需要。指定软件语句文件的路径。有关更多信息，请参阅 生成软件声明 (p. 36)。
agent.tags_folder	否	• /etc/dcv-session-manager-agent(Linux) • C:\Program Files\NICE\DCVSessionManagerAgent\conf\tags (Windows)	指定标记文件所在文件夹的路径。有关更多信息，请参阅 使用标签来定位 NICE DCV 服务器 (p. 17)。
agent.autorun_folder	否	• /var/lib/dcv-session-manager-agent/autorun(Linux) • C:\ProgramData\NICE\DcvSessionManagerAgent\autorun (Windows)	指定主机服务器上用于存储允许在会话启动时自动运行的脚本和应用程序的文件夹的路径。您必须指定绝对路径。必须可以访问该文件夹并且文件必须可由使用自动运行文件的请求参数创建会话API。
agent.max_virtual_sessions	否	-1 (没有限制)	使用 NICE DCV 会话管理器可以在 NICE DCV 服务器上创建的虚拟会话的最大数量。

参数名称	必填	默认值	说明
agent.max_concurrent_sessions_per_user	否	1	单个用户使用 NICE DCV 会话管理器可以在 NICE DCV 服务器上创建的虚拟会话的最大数量。
log.level	否	info	指定日志文件的详细程度等级。提供了以下详细程度等级： • error— 提供最少的详细信息。仅包括错误。 • warning— 包括错误和警告。 • info— 默认详细程度等级。包括错误、警告和信息消息。 • debug— 提供最多的详细信息。提供有助于调试问题的详细信息。
log.directory	否	• /var/log/dcv-session-manager-agent/(Linux) • C:\ProgramData\NICE\DCVSessionManagerAgent\log (Windows)	指定要在其中创建日志文件的目录。
log.rotation	否	daily	指定日志文件轮换。有效值为： • hourly— 日志文件每小时轮换一次。 • daily— 日志文件每天轮换一次。
log.max-file-size	否	10485760	当日志文件大小达到以字节为单位的指定大小时，它将被轮换。将创建一个新的日志文件，并在新文件中放置更多的日志事件。
log.rotate	否	9	轮换中保留的最大日志文件数。每次轮换并达到此数字时，最早的日志文件都将被删除。

NICE DCV 会话管理的发布说明和文档历史记录

本页面提供 NICE DCV 会话管理的发布说明和文档历史记录。

主题

NICE DCV 会话管理器发行说明

本节概述了 NICE DCV 会话管理器的主要更新、功能版本和错误修复。所有更新都按发布日期组织。我们经常更新文档以处理您发送给我们的反馈意见。

主题

2022.0-11952 年 —2022 年 2 月 23 日

内部版本号	更改和错误修复
• 代理：341 • 代理：520 • CLI：112	• 向 Agent 添加了日志轮换功能。 • 添加了配置参数以在 Broker 中将 Java 设置为主页。 • 改进了 Broker 中从缓存到磁盘的数据刷新。 • 修复了 CLI 中的 URL 验证问题。

2021.3-11591— 2021 年 12 月 20 日

内部版本号	新 功能
• 代理：307 • 代理：453	• 添加了对与 NICE DCV 连接网关集成的支持。

内部版本号	新 功能
• CLI：92	• 增加了对 Ubuntu 18.04 和 Ubuntu 20.04 的代理支持。

2021.2-11445— 2021 年 11 月 18 日

内部版本号	更改和错误修复
• 代理：288 • 代理：413 • CLI：54	• 修复了验证包含 Windows 域的登录名的问题。

2021.2-11190— 2021 年 10 月 11 日

内部版本号	更改和错误修复
• 代理：254 • 代理：413 • CLI：54	• 修复了命令行界面中无法启动 Windows 会话的问题。

2021.2-11042— 2021 年 9 月 1 日

内部版本号	新 功能	更改和错误修复
• 代理：254 • 代理：413 • CLI：37	• NICE DCV 会话管理器现在提供命令行界面 (CLI) 支持。您可以在 CLI 中创建和管理 NICE DCV 会话，而不是调用 API。 • NICE DCV 会话管理器引入了经纪商数据持久性 为了提高可用性，经纪商可以在外部数据存储中保存服务器状态信息，并在启动时恢复数据。	• 注册外部授权服务器时，您现在可以指定授权服务器用于签署 JSON 格式的 Web 令牌的算法。通过此更改，你可以使用 Azure AD 作为外部授权服务器。

2021.1-10557— 2021 年 5 月 31 日

内部版本号	新 功能	更改和错误修复
• 代理：214 • 代理：365	• NICE DCV 会话管理器添加了对传递给 Linux 上自动运行文件的输入参数的支持。 • 现在可以将服务器属性作为要求传递给创建会话API。	• 我们修复了 Windows 上自动运行文件的问题。

2021.0-10242— 2021 年 4 月 12 日

内部版本号	更改和错误修复

• 代理：183
• 代理：318
	• NICE DCV 会话管理器推出了以下新 API：
	  • 开放服务器
	  • 关闭服务器
	  • DescribeServers
	  • 获取会话截图
	• 它还引入了以下新的配置参数：
	  • 代理参数：`session-screenshot-max-width`、`session-screenshot-max-height`、`session-screenshot-format`、`create-sessions-queue-max-size`, 和`create-sessions-queue-max-time-seconds`.
	  • Agent 参数：`agent.autorun_folder`、`max_virtual_sessions`, 和`max_concurrent_sessions_per_user`.

2020.2-9662— 2020 年 12 月 4 日

内部版本号	更改和错误修复
• 代理：114 • 代理：211	• 我们修复了阻止经纪商启动的自动生成的 TLS 证书的问题。

2020.2-9508— 2020 年 11 月 11 日

内部版本号	更改和错误修复
• 代理：78 • 代理：183	• NICE DCV 会话管理器的初始版本。

文档历史记录

下表介绍此版本的 NICE DCV 会话管理的文档。

更改	描述	日期
不错的 DCV 版本 2022.0	NICE DCV 会话管理器已针对 NICE DCV 2022.0 进行了更新。有关更多信息，请参阅 2022.0-11952 年 —2022 年 2 月 23 日 (p. 52)。	2022年 2 月 23 日

更改	描述	日期
不错的 DCV 版本 2021.3	NICE DCV 会话管理器已针对 NICE DCV 2021.3 进行了更新。有关更多信息，请参阅 2021.3-11591— 2021 年 12 月 20 日 (p. 52)。	2021 年 12 月 20 日
不错的 DCV 版本 2021.2	NICE DCV 会话管理器已针对 NICE DCV 2021.2 进行了更新。有关更多信息，请参阅 2021.2-11042— 2021 年 9 月 1 日 (p. 53)。	2021 年 9 月 1 日
不错的 DCV 版本 2021.1	NICE DCV 会话管理器已针对 NICE DCV 2021.1 进行了更新。有关更多信息，请参阅 2021.1-10557— 2021 年 5 月 31 日 (p. 53)。	2021 年 5 月 31 日
不错的 DCV 版本 2021.0	NICE DCV 会话管理器已更新为与 NICE DCV 版本 2021.0 兼容。有关更多信息，请参阅 2021.0-10242— 2021 年 4 月 12 日 (p. 54)。	2021 年 4 月 12 日
NICE DCV 会话管理的初始版本。	此内容的第一版。	2020 年 11 月 11 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。