



Integrating Microsoft Teams with Pexip Infinity

Deployment Guide

Software Version 33

Document Version 33.a

October 2023

] pexip[

Contents

Integration overview	6
Integration features	6
VTC user experience when joining a Teams conference	6
Admitting guest participants from the lobby	8
Microsoft Teams Rooms SIP/H.323 calling	10
Using SIP Guest Join	10
Joining from a meeting room	11
Joining from a personal video endpoint (that is set up for One-Touch Join)	11
Audio participant avatars	11
Raised hand indicators	11
Recording or transcribing a Teams conference	12
Teams-like layout and Large Gallery view	12
Migrating to Microsoft Teams from Skype for Business	13
Streaming to Teams live events	14
Planning, prerequisites and firewall ports	15
Architecture overview	15
Pexip Infinity platform	15
Deployment and upgrade strategy	17
Preparing your Azure environment, regions and capacity planning	18
Network and certificate requirements	20
Obtaining and preparing the TLS certificate for the Teams Connector	21
Ensuring Conferencing Nodes have suitable certificates	22
Firewall ports for the Teams Connector and NSG rules	22
Additional deployment information	25
Certificate and DNS examples for a Microsoft Teams integration	26
Example deployment scenario	26
Teams Connector certificate requirements	27
Optional: internal/enterprise-based Conferencing Node guidelines	28
Public DMZ / Edge Conferencing Node guidelines	29
Migrating from — or co-existing with — a Skype for Business environment	31
Installing and configuring the Teams Connector in Azure	32
Prepare your Azure environment, certificates and other prerequisites	32
Download the Teams Connector application software files	32
Install the Teams Connector application into Azure	33
Specify installation variables used in the PowerShell commands	34
Enabling the CIS STIG image (GCC High / Azure US Government Cloud deployments only)	43
Installation commands to connect to Microsoft Graph, Azure Resource Manager and deploy the Teams Connector	43

Update DNS with Teams Connector name and IP address	52
Authorize Pexip CVI application to join Teams meetings	53
Authorize CVI app to bypass Teams lobby and configure dialing instructions	55
Next steps	58
Installing the Teams Connector using a blue-green deployment/upgrade strategy	59
Benefits of using a blue-green strategy	59
How it works	59
Deployment requirements and guidelines	60
Installing the Teams Connectors	61
Post deployment steps	63
Pexip Infinity Management Node configuration	64
Initial installation	64
Upgrade and test	64
Switching over to the new Teams Connector software	64
Upgrading the Teams Connector	65
Installing Teams Connectors in multiple Azure regions	66
Installing the additional Teams Connector application into Azure	66
Pexip Management Node configuration	68
Using certificate-based authentication for the Teams Connector CVI application	71
Deploying a new (first time) Teams Connector with certificate-based authentication	71
Upgrading or redeploying a Teams Connector that is already using CBA	76
Migrating (upgrading) an existing Teams Connector using password-authentication to CBA	79
Using private routing with the Teams Connector	81
Deployment environments and inter-network connectivity options	81
Requirements	81
Enabling private routing	81
Setting up a private DNS zone in Azure	83
Setting up peering where the Pexip Infinity platform is in Azure	84
Configuring Pexip Infinity as a Microsoft Teams gateway	86
Prerequisites	86
Ensuring a Microsoft Teams license is installed	86
Configuring global settings	86
Configuring your Microsoft Teams tenants	86
Configuring your Teams Connector (addresses, Event Hub, minimum capacity)	87
Configuring a Teams Connector per location	89
Configuring Virtual Receptions and Call Routing Rules for Teams meetings	89
Routing indirectly via a Virtual Reception (IVR gateway)	89
Call Routing Rules for direct and indirect routing	91

Enabling Microsoft Teams Rooms SIP/H.323 calling	96
Licensing and software requirements	96
Enabling your organization for Teams Room calling	96
Enabling VTCs to call a Microsoft Teams Room	97
Enabling a Microsoft Teams Room to call a VTC endpoint or VMR	99
Enabling SIP Guest Join	100
Enabling guest join in Pexip Infinity	101
Controlling the layout seen by VTC participants	102
Setting the default layout	102
Dynamically changing the layout during a conference	103
Using Adaptive Composition in a Teams conference	105
Customizing and enabling the in-lobby and mute notifications	105
Example themes	106
DNS and ports requirements	106
Interoperability and deployment features	106
Viewing Teams Connector instance, call and participant status	108
Viewing the status of Teams Connector instances	108
Monitoring calls placed into Teams conferences	108
Scheduling scaling and managing Teams Connector capacity	110
Using scheduled scaling	110
How scheduled scaling works	110
Setting the minimum number of instances	111
Setting the maximum number of instances	111
Configuring scheduled scaling policies	112
Manual scaling via the Azure portal	113
Maintaining your Teams Connector deployment	115
Modifying the Teams Connector's configuration	115
Upgrading the Teams Connector to the latest software	115
Using CBA with version 33	116
Redeploying the Teams Connector	117
Updating the Teams Connector TLS certificate or the CVI app certificate	122
Assigning Owner permissions for the API app	124
Maintaining access to the Admin Consent web page	124
Reinstating the Admin Consent web page	124
Changing the workstation / management network addresses that can access the consent page	125
Adding or removing Conferencing Nodes	125
Changing the alternative dialing instructions or IVR address	125
Changing the call capacity of a Teams Connector	126
Changing management workstation access	126
Mandating the use of Availability Zones	127
Monitoring the Teams Connector	127

Reviewing your log analytics workspace status and metrics	128
Enabling diagnostic logs	130
Uninstalling the Teams Connector	133
Troubleshooting Microsoft Teams and Pexip Infinity integrations	135
Installation issues	135
Call failures (invalid conference ID, rejected calls and disconnections)	137
Useful PowerShell commands	140
Obtaining Teams Connector logs	141
Data stored/processed by the Teams Connector	144
Discovering your Azure tenant ID	144
Viewing your tenant's app registrations	145
Viewing current app registrations	145
View authorized enterprise apps	146
Azure notification: Denial of service attack detected	146

Integration overview

Pexip Cloud Video Interoperability (CVI) enables professional SIP and H.323 video conferencing systems to join Microsoft Teams conferences as if they were native Microsoft clients.

It enables any video conferencing system to join Microsoft Teams meetings and allows authenticated systems to join as trusted participants without additional user interaction (i.e. lobby by-pass), including:

- H.323 & SIP room-based videoconferencing systems, including Cisco, Polycom, Lifesize, and others
- Browser-based video (WebRTC)

Third-party systems can connect to Teams meetings via the Infinity Gateway either via a Virtual Reception (IVR) or by dialing the conference directly.

You can also:

- Enable your Microsoft Teams Room systems to make and receive calls with SIP and H.323 endpoints, and join Pexip Infinity VMRs.
- Join Microsoft Teams meetings where that meeting is being hosted by an external third-party organization (even if the host's organization has not enabled Pexip interoperability themselves) — this is referred to as "SIP Guest Join". Note that this feature works by routing the call to the external organization via the Pexip Service.

Integration features

Pexip Infinity is a Microsoft-certified video interoperability platform for Microsoft Teams. The Pexip Teams Connector is a Pexip application that is deployed in Microsoft Azure and is used to enable Cloud Video Interoperability (CVI) with Microsoft Teams. It handles all Teams communications and meeting requests from the Pexip Infinity platform and passes them on to the Microsoft Teams environment.

The key features of Pexip's CVI integration with Microsoft Teams are:

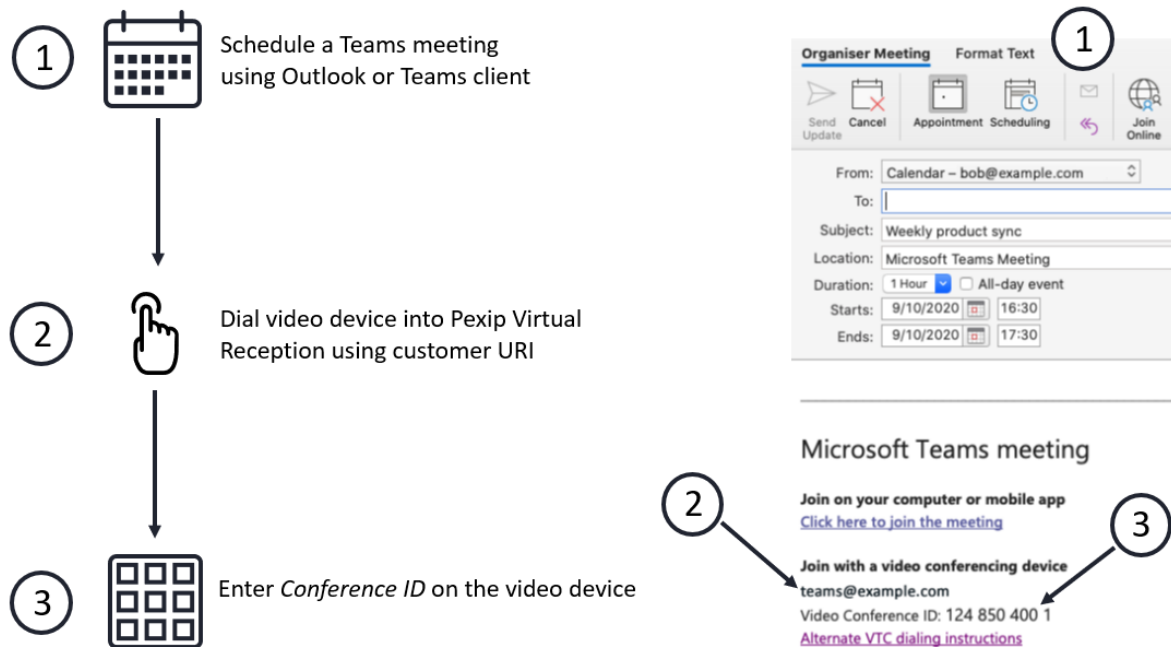
- Call in to a Teams meeting from any standards-based VTC system on SIP and H.323
- Lobby bypass for trusted endpoints, notifications and customizable waiting screen for untrusted VTC endpoints that are held in the lobby
- Choice of layouts for VTC participants, and spotlight support
- Audio-only participants are represented by their avatar from Exchange Online
- Raised hand, recording and transcription indicators
- Microsoft Teams Rooms SIP/H.323 calling (incoming and outgoing calls)
- Scheduled scaling to automatically scale up and down your call capacity at different times of the day
- Control and ownership of data
- Native VTC network resiliency and firewall traversal between private and public networks
- Bi-directional content sharing between VTCs and Microsoft Teams via Video-based Screen Sharing (VbSS)
- Synchronized mute/unmute control and status for VTCs
- Native scheduling via Outlook and Microsoft Teams client; join information is automatically added
- Tailored meeting invites with a customer-specific domain
- Full lobby and roster list control in the Teams client

Note that Microsoft Teams is inherently a dial-in service i.e. you can only dial **from** a third-party video system into Teams. You cannot dial **out** from Teams to a SIP, H.323 device etc — instead, you have to send the relevant joining instructions/invitation to the user of that device.

VTC user experience when joining a Teams conference

All VTC-based participants (SIP and H.323 devices) can either access the Teams conference via a customizable Virtual Reception service which prompts them to enter the Conference ID of the conference they want to join, or they may also be able to dial an address that takes them directly into a specific conference. Other software-based clients such as Skype for Business or Pexip's own Connect apps can also join via direct dial or via a Virtual Reception.

Pexip Infinity's deployment model allows the use of a customer-specific domain such as `teams@example.com` for dialing the Teams Virtual Reception.

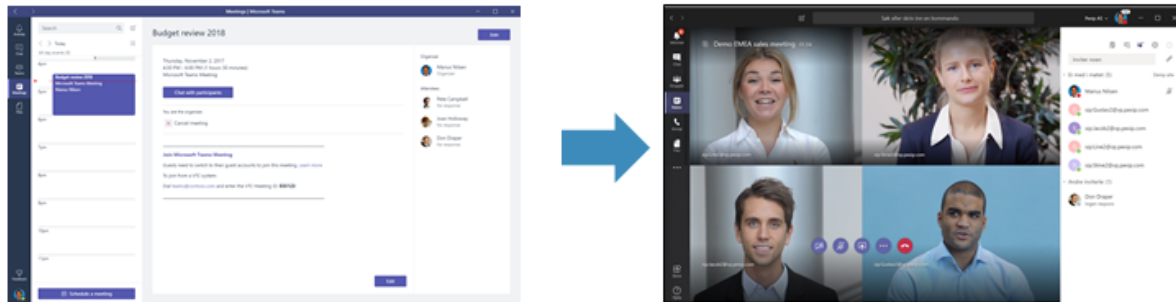


Alternative VTC dialing instructions can be provided that are customized to the company network and workflow such as:

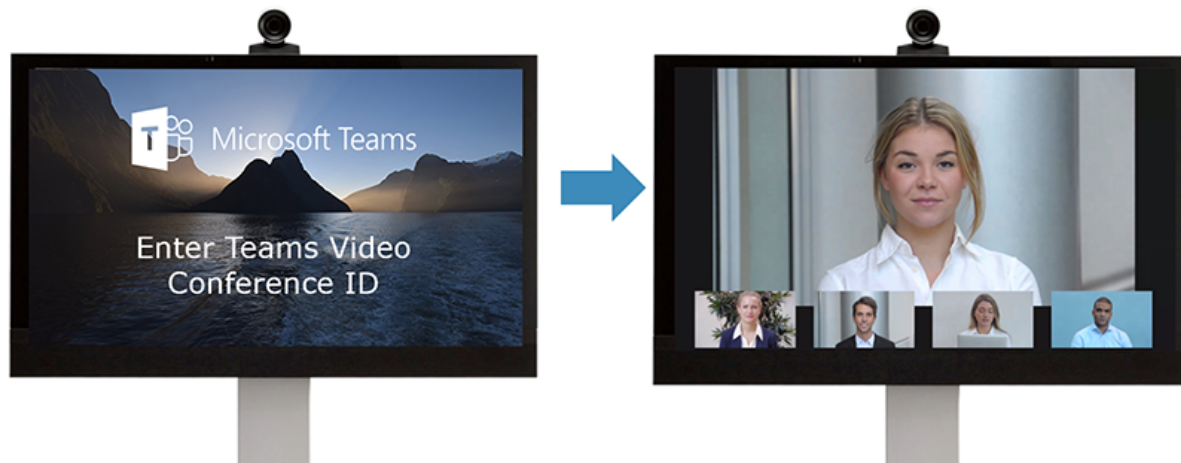
- 123958530@vc.example.com
- 123958530@104.215.95.187
- 104.215.95.187##123958530

Participants using a Teams client join a Teams meeting as usual, and any gatewayed third-party participants can be seen and heard in the same way as any other directly-connected Teams clients in that meeting. VTC participants see Pexip's standard 1+7 layout by default, but this can be changed to use other layouts, including a [Teams-like layout](#).

Authenticated, trusted VTCs that are located within the organization can join the conference directly, without any additional user interaction, whereas unauthenticated, untrusted external VTCs are admitted via the Teams lobby.



Join workflow from a Microsoft Teams client into a Teams meeting



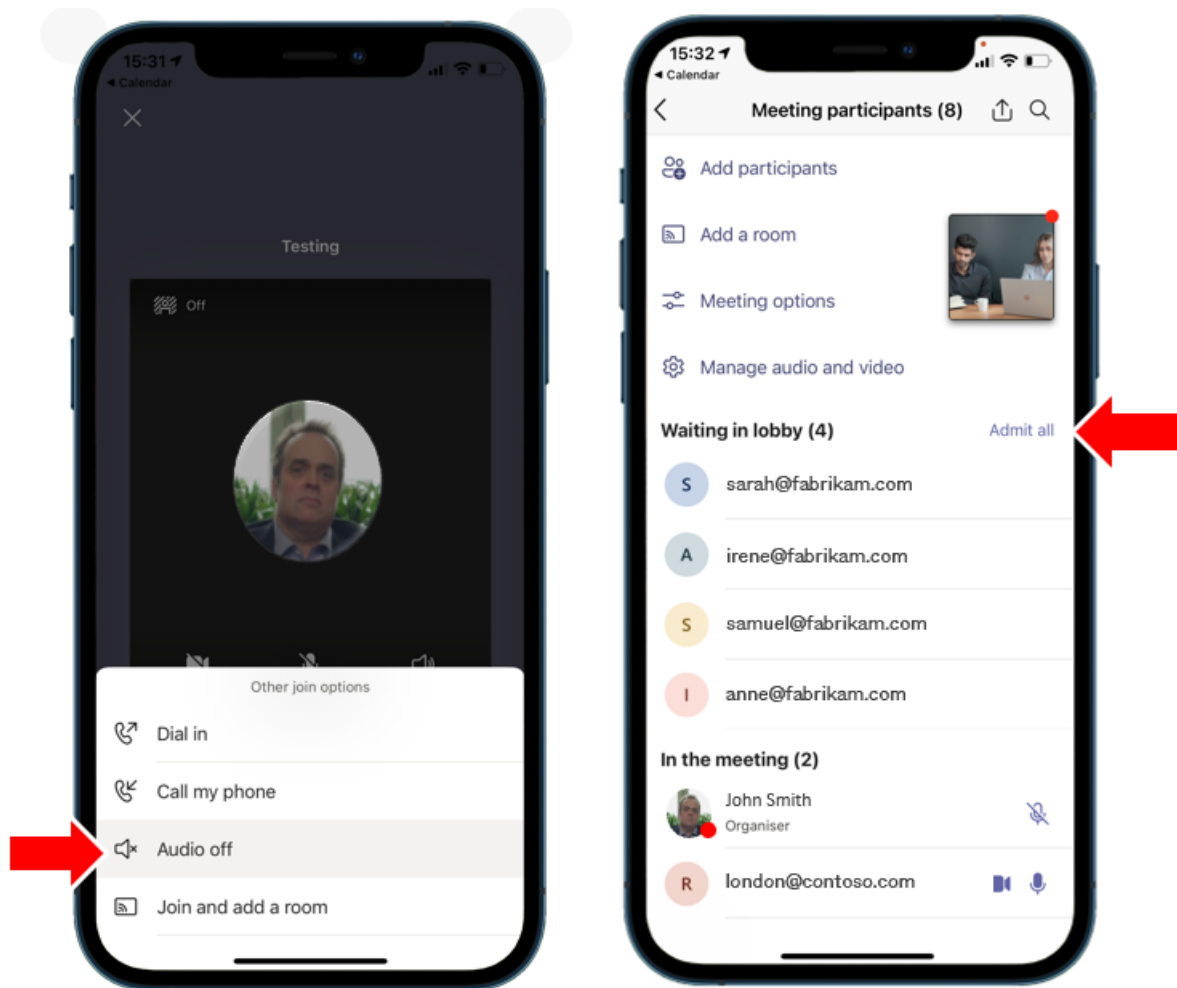
Join workflow from a third-party VTC system into a Microsoft Teams meeting (Pexip's default 1+7 layout)

Admitting guest participants from the lobby

Any VTC participants that are already in the conference are notified when an untrusted VTC or guest Teams client enters the Teams lobby. They see an on-screen message and hear a knock-knock sound.



The meeting host can then use their Teams client to admit the waiting guests (the VTC system cannot admit guests).



If only some guests are admitted, the VTC's display refreshes the count of waiting users. If all guests are admitted, the VTC's display briefly shows a "The lobby is empty" message.



Microsoft Teams Rooms SIP/H.323 calling

Your Microsoft Teams Room devices with Pro licensing can make and receive 1:1 (also referred to as point-to-point) SIP/H.323 video calls with VTCs.

Teams Room SIP/H.323 calling supports:

- Dialing from a VTC (internal or external) into a Teams Room.
- Dialing from a Teams Room to an internal or external VTC.
- Dialing from a Teams Room into a third-party SIP or H.323 enabled meeting, or a Pexip Infinity VMR (including PIN support).
- Bi-directional screensharing (BFCP or H239) between the Teams Room and the VTC or VMR.
- Sending DTMF tones from a Teams Room.

Note that:

- Different calling policies can be applied to certain rooms to, for example, not allow incoming SIP/H.323 calls, while other systems are enabled for this functionality depending on your business requirements.
- When joining a Teams Meeting the full feature set of a Teams Room is available, including participant list and other advanced meeting interaction features. However, this is **not** a call to a Teams meeting, thus when calling a SIP/H.323 destination many of the standard Teams meeting features (such as call transfer, share whiteboard, and add participant) are not available, as the functionality for 1:1 calls is focused around bidirectional audio/video communication and content sharing.

Using SIP Guest Join

Pexip's standard Microsoft Teams interoperability solution allows your own video conferencing endpoints (and your guests) to join Microsoft Teams meetings that you are hosting. The "SIP Guest Join" feature lets you join Microsoft Teams meetings with your own

video conferencing endpoints where that meeting is being hosted by an external third-party organization who has not enabled Pexip interoperability themselves.

If you are invited to an externally-hosted Teams meeting, you can join it via One-Touch Join on your video conferencing endpoint.

Joining from a meeting room

1. Invite the room (as a room resource) to the meeting. Either:
 - Copy the contents of the meeting invite, and use that to book the room (in this case the accept/reject message from the room is **sent to you**); or
 - Forward the meeting invite to the room's mailbox (in this case the accept/reject message is **sent to the meeting organizer**).
2. At the start time of the meeting: enter the room and press the Join button.

Joining from a personal video endpoint (that is set up for One-Touch Join)

1. Accept the meeting invite as usual (OTJ only looks for "accepted" meetings).
2. Press Join at the start of the meeting.

See [Enabling SIP Guest Join](#) for configuration and setup details.

Audio participant avatars

Teams users who are logged in and have joined via audio-only are represented by an avatar (supplied by Exchange Online) within the VTC's conference layout. If no avatar is available, or it is a guest user from another organization, a substitute graphic is generated based on the participant's initials.



Raised hand indicators

Notifications are shown to any VTC participants if a Teams participant raises their hand. An indicator slides out over the conference layout that identifies the participant with the raised hand and remains visible until the participant's hand is lowered.

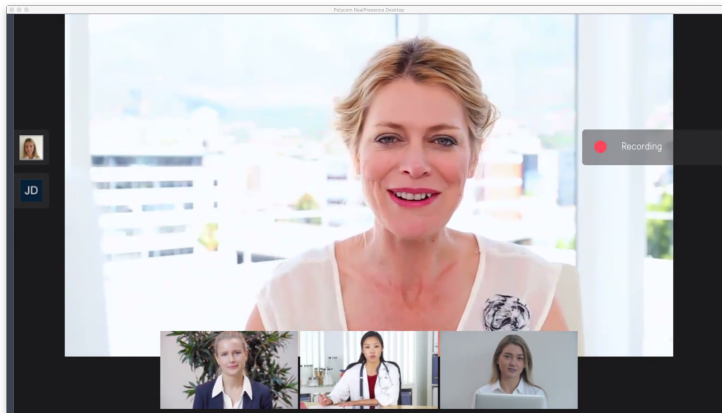


Up to 3 indicators may be displayed, in the order that the hand was raised, with the oldest at the top. If there are more than 3 hands raised then the third indicator also shows the number of additional raised hands.

Note that you cannot use the Connect apps to lower the hand of a Teams participant.

Recording or transcribing a Teams conference

If a Microsoft Teams conference is recorded or transcribed, relevant audio prompts indicating that recording/transcribing has been started/stopped are played to callers who are gatewayed via Pexip Infinity into the conference, and a recording/transcribing indicator is displayed.

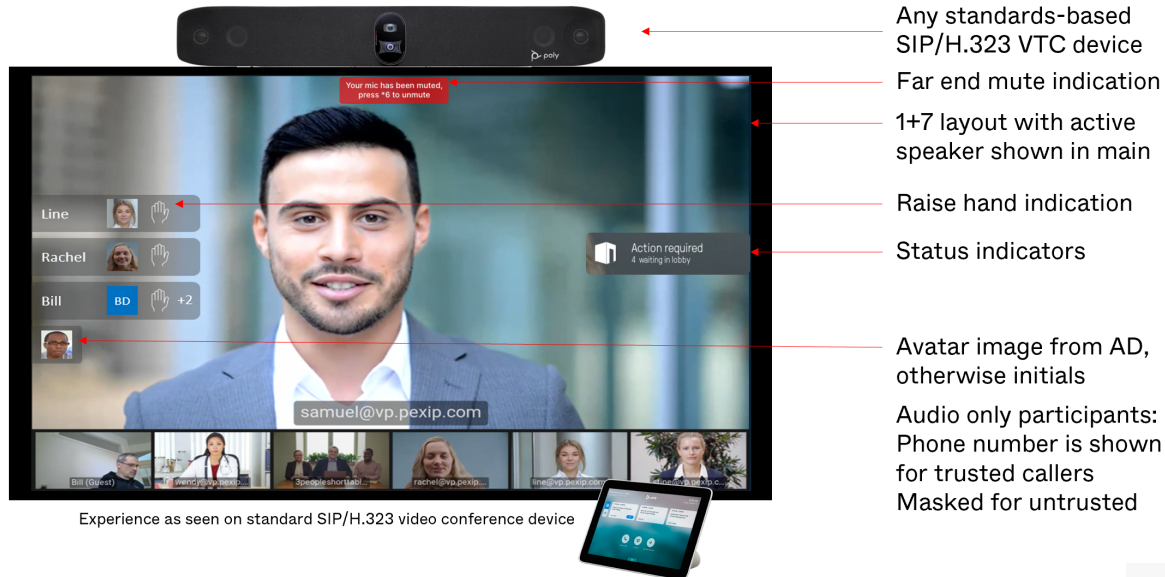
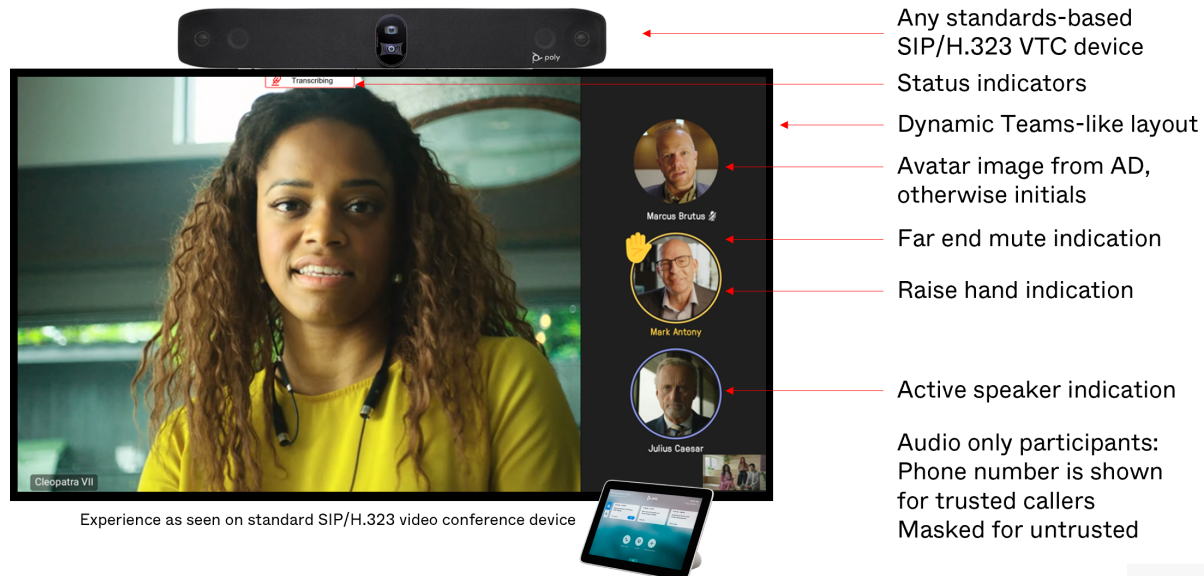


Teams-like layout and Large Gallery view

Pexip offers a Teams-like layout for VTC systems that looks and operates in a similar manner to the meeting experience seen by native Teams clients. In this layout, spotlighting, use of avatars for audio-only participants, active speaker indication and raised hand indicators all have a similar look and feel to the native Teams layout.

The Teams-like layout also supports multiscreen participant display, and receiving the presentation stream as part of the layout mix.

A comparison of the user experience between the 1+7 layout and the Teams-like layout is shown below.

CVI meeting with the 1+7 layout**CVI meeting with the Teams-like layout**

Pexip also supports Microsoft's Large Gallery view — you can use a customized theme to toggle between the Pexip layouts and Large Gallery view.

See [Controlling the layout seen by VTC participants](#) for more information about layouts.

Migrating to Microsoft Teams from Skype for Business

Pexip Infinity works simultaneously with both Microsoft Teams and Skype for Business. This means that users can be enabled to use both platforms and they can be migrated from one platform to the other at your own pace. Interoperability into either platform is handled by the same single Pexip Infinity installation, and the same Conferencing Nodes.

For example you could provide a dial-in lobby address of:

- `skype@example.com` for interoperability into Skype for Business meetings
- and
- `teams@example.com` for interoperability into Teams meetings

and then in each case the user would be directed to the appropriate Pexip Virtual Reception and would enter the appropriate conference ID for the relevant Microsoft meeting platform.

Microsoft Teams interoperability with Skype for Business

Microsoft Teams supports some level of interoperability with Skype for Business. This includes the ability to do a point-to-point voice and video call between Microsoft Teams and Skype for Business (depending on SfB/Teams setup, see <https://docs.microsoft.com/en-us/microsoftteams/teams-and-skypeforbusiness-coexistence-and-interoperability#native-interop-experiences>).

As Pexip supports native Skype for Business federation, it is technically possible to place a call between a Teams client (if the tenant is set up in a supported mode for SfB federation) and Pexip. However due to the [limitations](#) of this interoperability, in particular the lack of support for screen sharing / app sharing between Microsoft Teams and Skype for Business, this is not a use case that Pexip recommends or provides support for.

The certified interoperability for Microsoft Teams is for a Teams meeting to be scheduled (by an organization enabled for Pexip interoperability), and non-Teams compatible systems to dial in to the Teams meeting (see [user experience](#) below). This type of integration allows dual stream content sharing.

Streaming to Teams live events

You can use Pexip Infinity as an external RTMP encoder for your Teams live events, which means that you can stream video, audio and presentation content from your videoconferencing meeting room systems directly into the event.

Planning, prerequisites and firewall ports

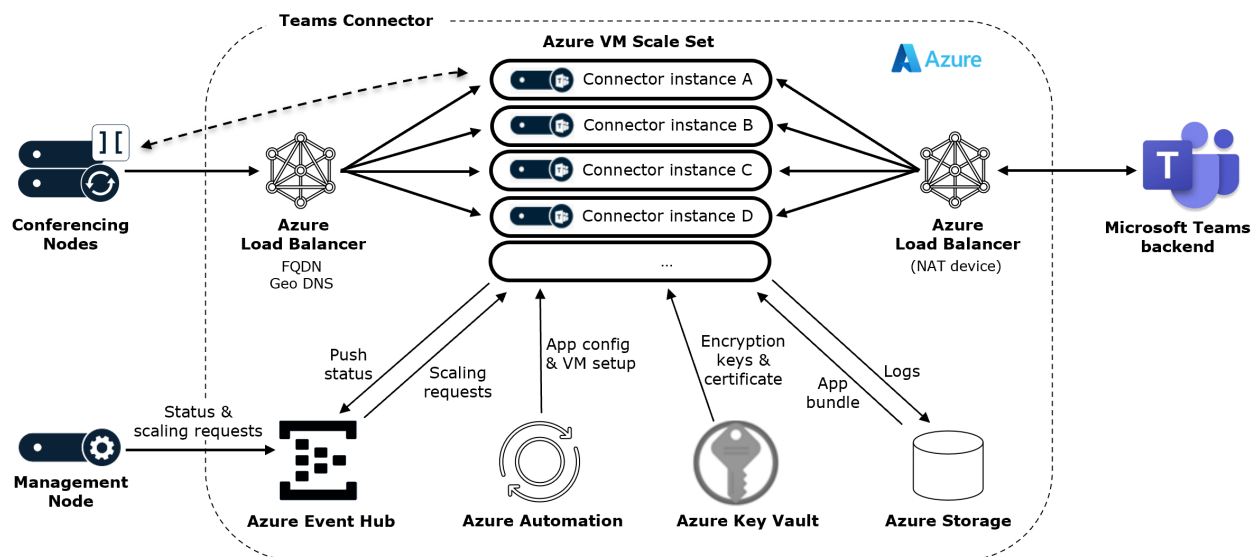
Architecture overview

The Pexip Teams Connector is a Pexip application that is deployed in Microsoft Azure and is used to enable Cloud Video Interoperability (CVI) with Microsoft Teams. It handles all Teams communications and meeting requests from the Pexip Infinity platform and passes them on to the Microsoft Teams environment. The dedicated application ensures control and ownership for organizations with stringent regulatory compliance requirements.

The diagram below shows the Teams Connector components that are deployed in Azure, and how they interact with the Pexip Infinity platform and Microsoft Teams. Note that:

- The Azure Virtual Machine scale set (VMSS) allows the Pexip application to run across a group of identical, load balanced VMs.
- The Azure Standard Load Balancer enables the use of Azure Availability Zones, which are used by default if they are available in your selected region. It is represented twice in the diagram (performing load balancing towards Pexip Infinity, and NAT towards Teams), but it is the same single Azure resource.

You do not have to set up these Azure components individually — they are all created as part of the Teams Connector deployment process.

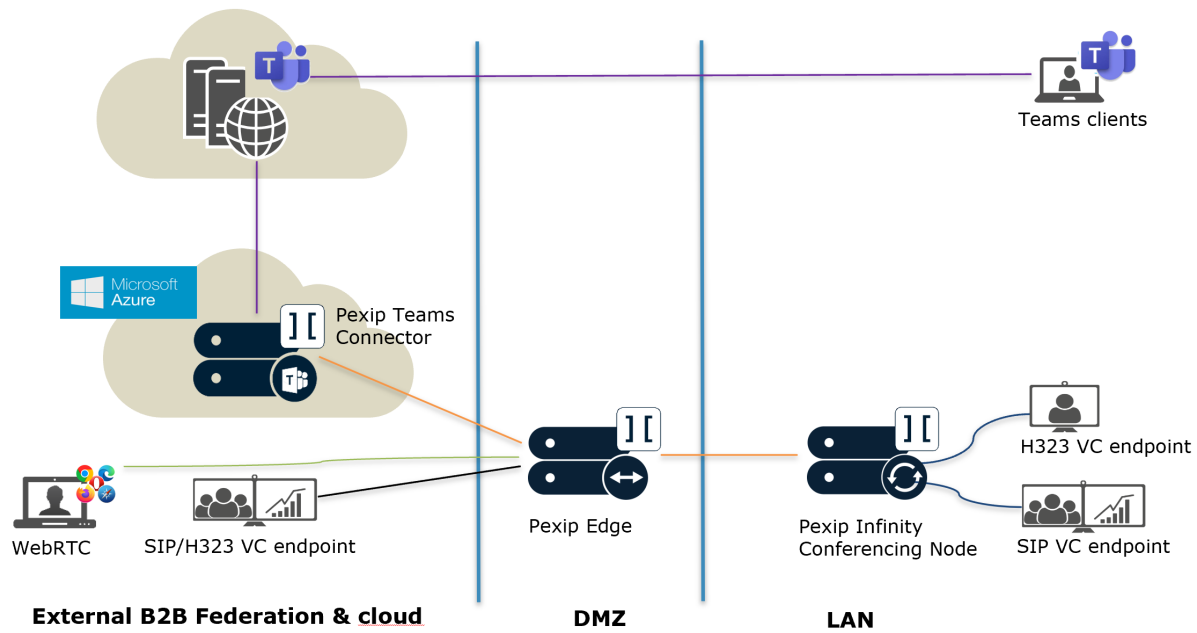


Pexip Infinity platform

While the Teams Connector must be deployed in Microsoft Azure, the Pexip Infinity platform can be installed in any supported environment such as on-premises or in a public or hybrid cloud (which would typically be Microsoft Azure when integrating with Microsoft Teams).

On-premises deployment

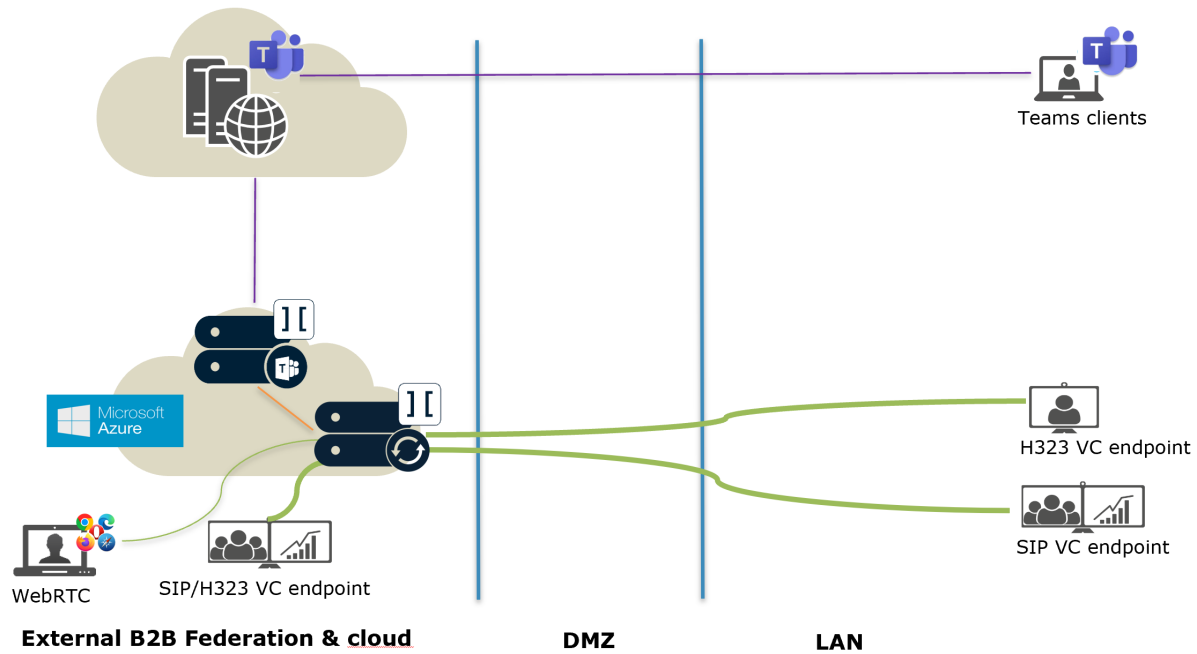
The Pexip Infinity platform can be deployed on-premises with public-facing Conferencing Nodes used to connect to the Pexip Teams Connector in Azure.



In this example deployment, external endpoints and federated systems, as well as on-premises devices can all connect to Teams conferences via the Pexip DMZ nodes.

Cloud-hosted deployment

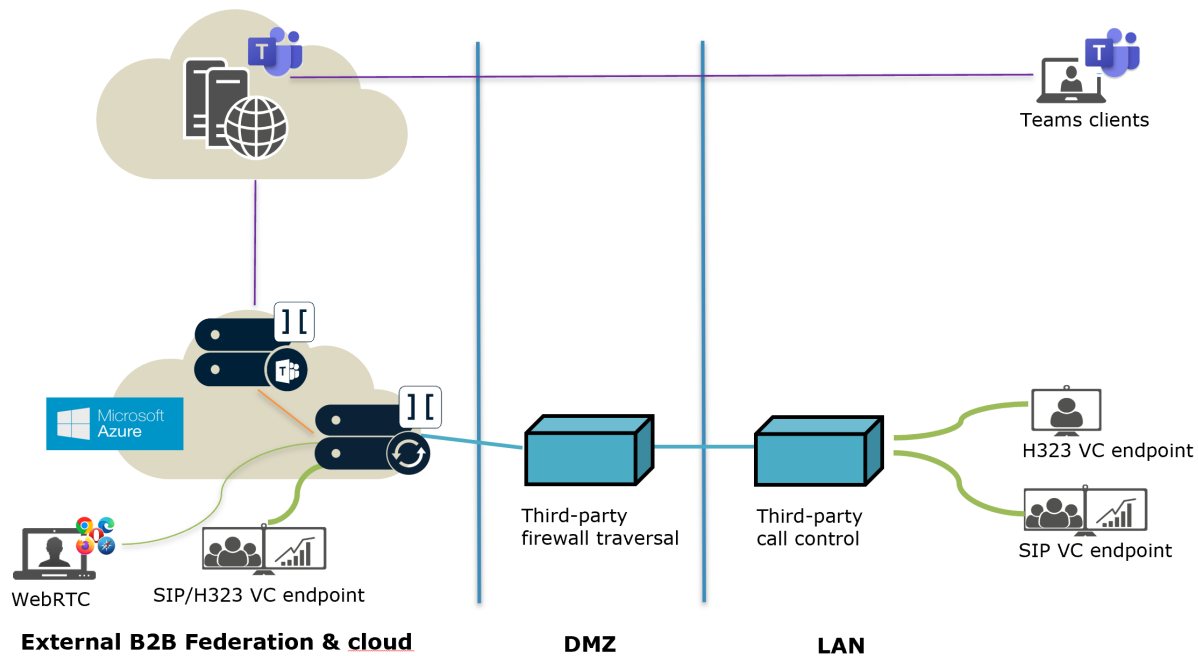
The Pexip Infinity platform can be deployed in a dedicated public or hybrid cloud within your own cloud subscription, providing full control over your environment.



Here, external endpoints, federated systems and on-premises devices can all connect to Teams conferences via the cloud-hosted Pexip Infinity nodes. You could use any supported cloud service but you would typically deploy your Conferencing Nodes in Microsoft Azure alongside your Pexip Teams Connector.

Including third-party call control

The Pexip Teams Connector and the Pexip Infinity platform can both be deployed in Azure with an on-premises, third-party call control system.



If you have a third-party call control system that you want to retain, it can be configured to connect your on-premises systems to the cloud-hosted Pexip Infinity platform.

Pexip Infinity has a close integration with Microsoft Teams and uses Teams APIs and Microsoft SDKs to provide Infinity's interoperability features. Even though Pexip strives to maintain backwards compatibility between older versions of Pexip Infinity and the latest release of Microsoft Teams, to ensure compatibility with the latest updates to Teams we recommend that you aim to keep your Pexip Infinity deployment up-to-date with the latest Pexip Infinity software release. If, for example, you have a large Pexip deployment for non-Teams related services, and you have stringent upgrade procedures meaning that you do not always keep your Infinity software up-to-date with the latest release, you may want to consider deploying a second instance of the Pexip Infinity platform that is dedicated to your Teams interoperability requirements, and which can be managed separately and upgraded more frequently.

Deployment and upgrade strategy

The Teams Connector deployment/upgrade process supports a blue-green deployment strategy. This approach allows you to create separate environments where one environment (blue) is running the current application version and another environment (green) is running the new application version. This means that you can test both of these deployments separately, and switch between them.

This is different from the regular deployment and upgrade strategy that involves deploying a single Teams Connector environment that is destructively replaced during the upgrade process.

We recommend using a blue-green deployment strategy, where you have two Teams Connector environments deployed in parallel, as it:

- Provides a non-destructive upgrade path.
- Increases application availability during the upgrade process.
- Enables upgrade activities to be done in business hours when access to required service administrators/owners are more readily available, before a planned "switch over" window.
- Reduces time-pressure and risk if there are delays due to Azure resources not being available.
- Reduces deployment risk by simplifying the rollback process if a deployment fails.

See [Installing the Teams Connector using a blue-green deployment/upgrade strategy](#) for full details, and see [this article](#) for more background information on blue-green deployments.

Preparing your Azure environment, regions and capacity planning

This section lists the various preparation steps you must perform before starting your Teams Connector installation into Azure.

Obtain an Azure subscription and an Azure tenant ID

Ensure that you have an Azure subscription and an [Azure tenant ID](#) for your Teams Connector deployment.

Note that some of the installation steps must be performed by somebody with **Owner** permissions for the Azure subscription (see [Azure permissions requirements](#) for more information).

Decide Azure deployment region(s) and check quota

Decide in which Azure region you want to deploy the Teams Connector. Large enterprises may want to install a Teams Connector in multiple regions.

- The Azure region must support Automation and Fs series instance types.

See [Azure automation](#) for more information about Automation, and [Azure product availability by region](#).

- Ensure that you have sufficient resource quota and capacity for your region and instance types.

By default, Azure Resource Manager virtual machine cores have a regional total limit **and** a regional per series limit, that are enforced per subscription. Typically, for each subscription, the default quota allows up to 10-20 CPU cores per region and 10-20 cores per series.

The allocated quota may be increased by opening a support ticket with Microsoft via the Azure Portal. Based on your capacity requirement, you should request a quota increase for your subscription. Ensure that you request a sufficient number of CPU cores. Each Teams Connector instance will use 4 vCPU of type Fs-series. Thus, for example, if 6 Teams Connector instances are required, then the quota must be increased to 4 cores x 6 Fs-series instances = 24 CPU cores of type Fs-series. However we strongly recommend that you request a quota covering more than the minimum, such as 40 cores, to allow for an increase in the future. It may take a number of days for the quota increase request to be processed. For more information see [this article](#).

Azure regions with Fs series instance type support

This is the list of Azure regions that support Fs series instances and Automation as of August 2023.

Theater	Region	RegionName
North America		
	Canada Central	canadacentral
	Canada East	canadaeast
	Central US	centralus
	East US	eastus
	East US 2	eastus2
	North Central US	northcentralus
	South Central US	southcentralus
	West Central US	westcentralus
	West US	westus
	West US 2	westus2
	West US 3	westus3

Theater	Region	RegionName
South America		
	Brazil South	brazilsouth
Europe		
	France Central	francecentral
	Germany West Central	germanywestcentral
	North Europe	northeurope
	Norway East	norwayeast
	Poland Central	polandcentral
	Sweden Central	swedencentral
	Switzerland North	switzerlandnorth
	UK South	uksouth
	UK West	ukwest
	West Europe	westeurope
APAC		
	East Asia	eastasia
	Japan East	japaneast
	Japan West	japanwest
	Korea Central	koreacentral
	Korea South	koreasouth
	Qatar Central	qatarcentral
	Southeast Asia	southeastasia
	UAE North	uaenorth
Australia		
	Australia Central	australiacentral
	Australia East	australiaeast
	Australia Southeast	australiasoutheast
India		
	Central India	centralindia
	South India	southindia
Africa		
	South Africa North	southafricanorth

You can use the following PowerShell script to list the current set of Azure regions that support Fs series (Standard_F4s) instances and Automation:

```
$($automationLocations = ((Get-AzResourceProvider -ProviderNamespace Microsoft.Automation).ResourceTypes | Where-Object ResourceTypeName -eq automationAccounts).Locations;
$vmLocations = ((Get-AzResourceProvider -ProviderNamespace Microsoft.Compute).ResourceTypes | Where-Object ResourceTypeName -eq locations/virtualMachines).Locations;
$vmInstanceLocations = $vmLocations | ?{ 'Standard_F4s' -in @(Get-AzVMSize -Location $_ | %{ $_.Name }) };
Get-AzLocation | ?{ ($_.DisplayName -in $automationLocations) -and ($_.DisplayName -in $vmInstanceLocations) }
) | Select-Object -Property DisplayName, Location
```

GCC High / Azure US Government Cloud deployments

You can deploy Teams Connector in the standard Azure commercial cloud, or where necessary for specific compliance requirements, in the Azure US Government environment.

Note that you can deploy Teams Connector in Azure US Government in combination with a commercial Teams environment or a Teams GCC environment. But, for Teams GCC High environments, Teams Connector must be deployed in Azure US Government.

- 📘 The Teams Connector must be hosted in the Arizona or Texas Azure regions for GCC High. The Virginia Azure region does not support the hosting of the Teams Connector for GCC High.

Most of the deployment processes are the same for either Azure environment, but there are some extra steps and minor differences when deploying in a US Government environment. Where there are variances indicated within this documentation you should either follow the **standard deployment** instructions (for commercial Azure) or the **GCC High / Azure US Government Cloud deployment** instructions.

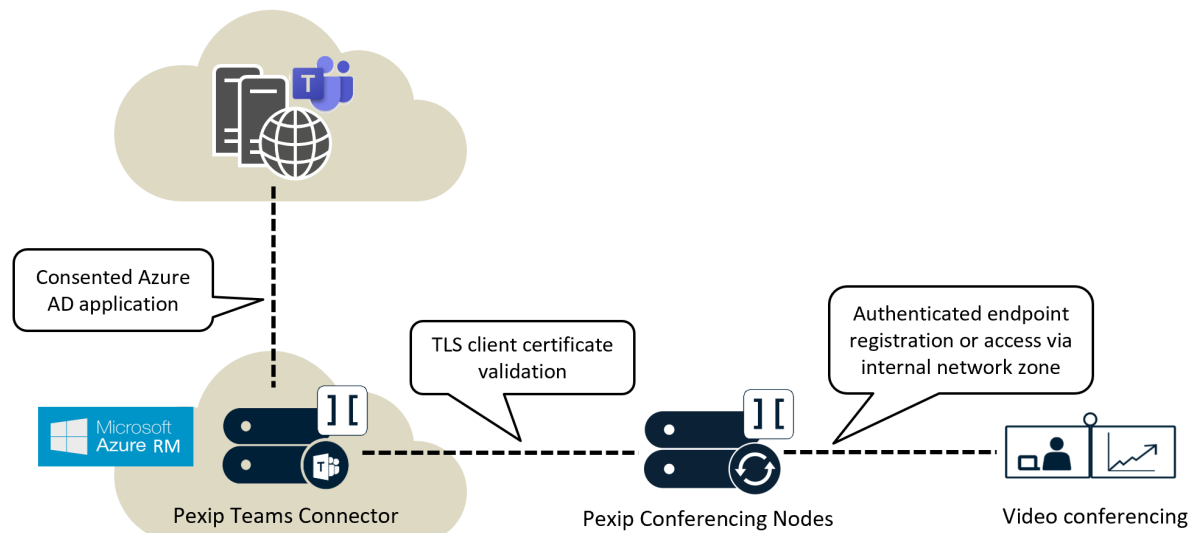
Capacity planning

Each Teams Connector instance can handle a maximum of 16 calls, although the capacity of each instance can vary depending on various factors such as call resolution, the number of presentation streams, and the number of participants in the Teams meetings. For capacity planning purposes we recommend that you assume 15 calls per instance.

- See [Scheduling scaling and managing Teams Connector capacity](#) for information about how you can control your available capacity.

Network and certificate requirements

This diagram shows how the main elements in a standard Microsoft Teams integration communicate with each other and how the connection between each element is validated/authenticated.

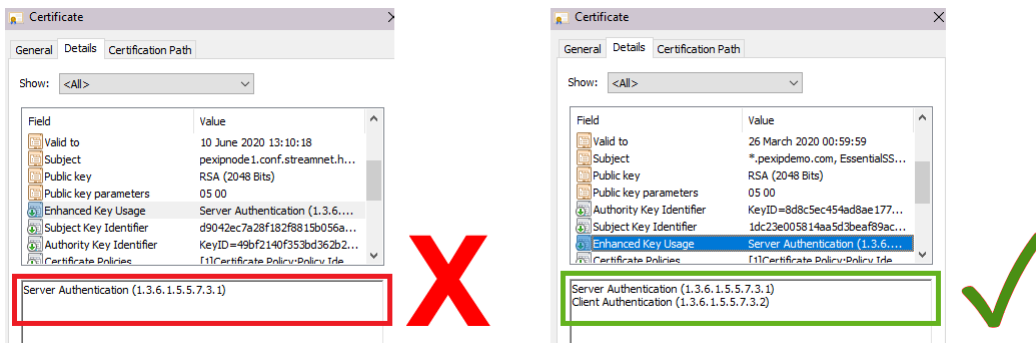


- You must have one or more publicly-reachable Conferencing Nodes. Those nodes:
 - can be Transcoding Conferencing Nodes or Proxying Edge Nodes
 - can have static NAT and/or dual network interfaces, as the Teams Connector is treated as a lineside connection.
- The public-facing Conferencing Nodes always communicate with the Teams Connector via public IP, even if they are within the same Azure tenant.
- The Teams Connector communicates with the Teams (Microsoft 365) backend via public IP; all traffic stays within the Microsoft network.
- The Teams Connector supports connections over TLSv1.2 only, and does not support RC2, RC4, DES and 3DES ciphers.

As an alternative network configuration you can consider using private routing* between the Teams Connector and Pexip Infinity — see [Using private routing with the Teams Connector](#) for more information.

In summary, the certificate usage principles are:

- The Teams Connector and Pexip Infinity validate the connection in both directions by TLS client certificate validation. This means that every certificate's Enhanced Key Usage properties must be set for both server **and** client authentication.



- Public-facing Conferencing Nodes must have a valid publicly-signed PEM-formatted certificate (typically with a .CRT or .PEM extension).
- The Teams Connector must have a publicly-signed PFX-formatted certificate. Multiple names/certificates are required if deploying Teams Connectors in several regions.

Obtaining and preparing the TLS certificate for the Teams Connector

You must install on the Teams Connector a TLS certificate that has been signed by an external trusted CA (certificate authority).

i You need to have this certificate available before you install the Teams Connector.

The certificate must be in Personal Information Exchange Format (PFX), also known as PKCS #12, which enables the transfer of certificates and their private keys from one system to another. It must use RSA keys.

- Decide on the FQDN (DNS name) you will use for the Teams Connector load balancer in Azure that will front the Teams Connector deployment e.g. `pexip-teamsconn-eu.teams.example.com`.
 - This FQDN is what you will use as:
 - the value of `$PxTeamsConnFqdn` in the [variables initialization script](#)
 - the certificate's subject name
 - the DNS name you will configure in Pexip Infinity (Call Control > Microsoft Teams Connectors > Address Of Teams Connector) later in the process.
 - It can use the same domain space as your Pexip Infinity deployment, or your Teams deployment, or it can use an altogether different domain. In all cases you always need to create the necessary DNS CNAME record(s) and public certificates for the chosen domain.
 - If you intend to deploy other Teams Connectors in other Azure regions, you will need a different DNS name for each Teams Connector and a certificate that matches that identity. You can use a single certificate for this, containing Subject Alternative Name (altNames attribute) entries for all of the regional Teams Connectors.

- It can be a wildcard certificate, where the wildcard character ('*') is the only character of the left-most label of a DNS domain name. Note that Pexip supports [RFC 6125](#) — this means that if you are using subdomains then, for example, a wildcard certificate of *.example.com would match foo.example.com but not bar.foo.example.com or example.com.
 - 2. Request a certificate for that name and generate the certificate in PFX format. Any intermediate certificates must also be in the PFX file.
- i** You can use the Pexip Infinity Management Node to generate a certificate signing request (CSR).
- i** You can use the Pexip Infinity Management Node to convert PEM certificates to PFX format (or vice versa), by uploading a PEM-formatted certificate and then downloading it again in PFX format. When downloading you can also include the private key and all necessary intermediate certificates in the PFX bundle.

Ensuring Conferencing Nodes have suitable certificates

The Conferencing Nodes (typically Proxying Edge Nodes) that will communicate with the Teams Connector must have TLS certificates installed that have been signed by an external trusted CA (certificate authority). If a chain of intermediate CA certificates is installed on the Management Node (to provide the chain of trust for the Conferencing Node's certificate) those intermediate certificates must not include any HTTP-to-HTTPS redirects in their AIA (Authority Information Access) section.

We recommend that you assign a "pool name" to all of the Conferencing Nodes that will communicate with the Teams Connector. The pool name should be used as a common Subject name on the certificate that is uploaded to each of those Conferencing Nodes. The certificate should also contain the individual FQDNs of each of the nodes in the pool as a Subject Alternative Name on the certificate. This pool name can then be specified on the Teams Connector (the \$PxNodeFqdns variable in the [initialization script](#)) as the name of the Conferencing Nodes that it will communicate with.

This approach makes it easier to add extra Conferencing Nodes into the pool as they will all present the same certificate/subject name to the Teams Connector. If you add a new Conferencing Node with a name that is not configured on the Teams Connector you will have to redeploy the Teams Connector and specify the new names.

See [Certificate and DNS examples for a Microsoft Teams integration](#) for more information and examples about certificates, DNS records and using a "pool name" for Conferencing Nodes.

Firewall ports for the Teams Connector and NSG rules

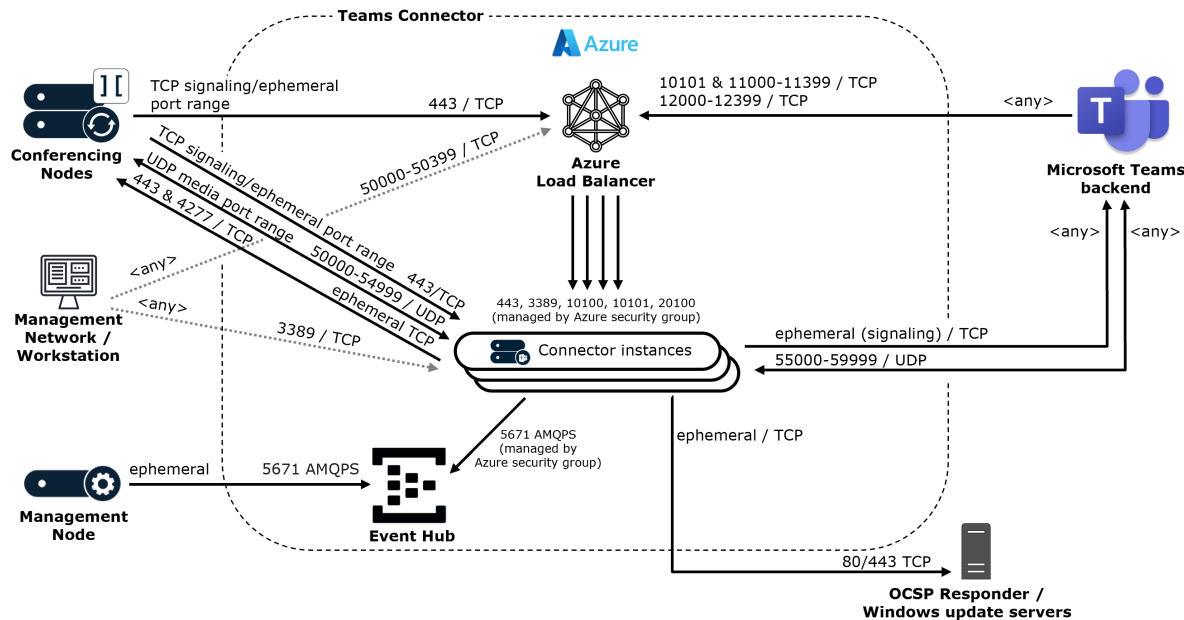
The following table lists the ports/protocols used to carry traffic between the Teams Connector components and Teams (Microsoft 365), your public-facing Conferencing Nodes (typically Proxying Edge Nodes), the Management Node and any management networks.

Source address	Source port	Destination address	Destination port	Protocol	Notes
Conferencing Nodes					
Conferencing Nodes	33000–39999 * ephemeral	Teams Connector load balancer Teams Connector instance	443	TCP	Initial call signaling is via the load balancer. When the call is established, signaling is directly between the Conferencing Node and the Teams Connector instance.
Conferencing Nodes	40000–49999 *	Teams Connector instance	50000-54999	UDP	Call media
Management Node					
Management Node	ephemeral	Teams Connector Azure Event Hub	5671	AMQPS	Only required if the Azure Event Hub is enabled for advanced status reporting
Teams Connector components					
Teams Connector instance	ephemeral	Teams (Microsoft 365)	<any>	TCP	Signaling

Source address	Source port	Destination address	Destination port	Protocol	Notes
Teams Connector instance	ephemeral	Conferencing Nodes	443 4277	TCP	Signaling (4277 is for calls between Microsoft Teams Rooms and VTC endpoints)
Teams Connector instance	50000-54999	Conferencing Nodes	40000–49999 *	UDP	Call media
Teams Connector instance	55000-59999	Teams (Microsoft 365)	<any>	UDP	Call media
Teams Connector instance	ephemeral	OCSP responder	80	TCP	Certificate revocation checking
Teams Connector instance	ephemeral	Windows update servers	80/443	TCP	Windows updates
Teams Connector instance	ephemeral	Teams Connector Azure Event Hub	5671	AMQPS	Only required if the Azure Event Hub is enabled for advanced status reporting; this port is managed by the Azure NSG
Teams (Microsoft 365)					
Teams (Microsoft 365)	<any>	Teams Connector load balancer	10101 11000-11399 12000-12399	TCP	Signaling
Teams (Microsoft 365)	<any>	Teams Connector instance	55000-59999	UDP	Call media
Management					
Management workstation	<any>	Teams Connector load balancer	50000-50399	TCP	Only enabled for any workstation addresses specified during Teams Connector installation
		Teams Connector instance	3389		
Client application viewing the meeting invitation					
<any>	<any>	Conferencing Nodes †	443	TCP	Access to Alternative Dial Instructions

* Configurable via the Media port range start/end, and Signaling port range start/end options.

† The Conferencing Nodes referenced in the `InstructionUri` for the "Alternate VTC dialing instructions".



Teams Connector Network Security Group (NSG)

A Network Security Group that supports these firewall requirements is created automatically in Azure as a part of the Teams Connector installation process, and is assigned to each Teams Connector instance. Note that the NSG includes:

- Rules used for internal traffic within the Teams Connector that is forwarded from the load balancer to the instances (to ports 10100 and 20100) — these ports do not need to be opened between the Conferencing Nodes / Microsoft Teams and the Teams Connector. Similarly, the NSG allows the instances to push data to the Event Hub.
- An "RDP" rule (priority 1000): if the `$PxMgmtSrcAddrPrefixes` installation variable contains addresses, this rule allows RDP access to the Teams Connector instances from those addresses. If no addresses are specified then a Deny rule is created (a full redeploy is required if you subsequently want to allow RDP access).

You may need to modify some of the NSG rules in the future if you subsequently [add more Conferencing Nodes](#) to your Pexip Infinity platform, or [change the addresses of any management workstations](#).

Inbound security rules

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION
100	Teams-signalling-endpoint	10100	TCP	Any	Any	✓ Allow ...
120	MCU-signalling-endpoint	443	TCP	46.19.20.105,81.14...	Any	✓ Allow ...
130	Skype-MediaProcessor-NMF-endp...	20100	TCP	Any	Any	✓ Allow ...
140	Media-ports	50000-59999	UDP	Any	Any	✓ Allow ...
1000	RDP	3389	TCP	46.19.20.98,62.25...	Any	✓ Allow ...
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	✓ Allow ...
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	✓ Allow ...
65500	DenyAllInBound	Any	Any	Any	Any	✗ Deny ...

Outbound security rules

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION
65000	AllowVnetOutBound	Any	Any	VirtualNetwork	VirtualNetwork	✓ Allow ...
65001	AllowInternetOutBound	Any	Any	Any	Internet	✓ Allow ...
65500	DenyAllOutBound	Any	Any	Any	Any	✗ Deny ...

In addition to this:

- You must allow the relevant ports through any of your own firewalls that sit between the Teams Connector components and your public-facing Conferencing Nodes and management networks.
- If you enable [advanced status reporting](#) you must also allow the Pexip Infinity Management Node to connect out to the Azure Event Hub component. If required you can also lock down the Event Hub to only allow internet access from your Management Node IP address; access is currently unrestricted as it is secured by a connection string and the AMQPS protocol.

Additional deployment information

The following features are provided/enabled automatically as part of the deployment process:

- VMSS (virtual machine scale set) disk encryption is enabled by default. The keys are stored in an Azure Key Vault. Note that the disk encryption can affect performance for approximately 30 minutes after the deployment has finished.
- The Teams Connector VMs use managed identities to access the storage account in the static resource group that is used for logging, crash reports and the application ZIP.

Certificate and DNS examples for a Microsoft Teams integration

This topic contains example Conferencing Node naming patterns, certificate, and DNS requirements for an integration with Microsoft Teams. You can use this example as the basis for your own integration, changing the example domain and DNS names as appropriate for your particular environment.

- [Example deployment scenario](#)
- [Teams Connector certificate requirements](#)
- [Optional: internal/enterprise-based Conferencing Node guidelines](#)
- [Public DMZ / Edge Conferencing Node guidelines](#)
- [Migrating from — or co-existing with — a Skype for Business environment](#)

Example deployment scenario

In this example deployment scenario:

- The Pexip Infinity platform is on the **vc.example.com** VTC subdomain. It has (optional) privately-addressed Conferencing Nodes on the enterprise network, and (mandatory) publicly-reachable Conferencing / Edge Nodes in the public DMZ.
- The Pexip Teams Connector is a publicly-reachable application in Microsoft Azure at **pexip-teamsconn-eu.teams.example.com**.
- Externally-located SIP devices, H.323 devices and Pexip Connect apps can call into the platform to be routed into Microsoft Teams conferences (or other Pexip Infinity services) using aliases in the format **<alias>@example.com** and/or **<alias>@vc.example.com**.

Your choice of routing via your primary domain (**example.com** in this case) and/or via your Pexip subdomain (**vc.example.com**) depends on your existing environment and dial plan requirements. For example, if your environment already includes DNS records and routing for SIP and H.323 devices to your primary domain for other purposes, then you can use the Pexip subdomain (**vc.example.com**) as your interoperability route into Teams for all call protocols.

Routing via your primary domain is preferable as it allows for simpler dial-in addresses such as **teams@example.com** (see example invitation, right).

- Federated Skype for Business / Lync clients can also call into the platform using aliases in the format **<alias>@vc.example.com**.
- Enterprise-based SIP devices, H.323 devices and Pexip Connect apps can call into Microsoft Teams conferences (or other Pexip Infinity services) using aliases in the format **<alias>@example.com** and **<alias>@vc.example.com**.

The following diagram shows the example Conferencing Node naming patterns, certificate, and DNS requirements to support these call scenarios.

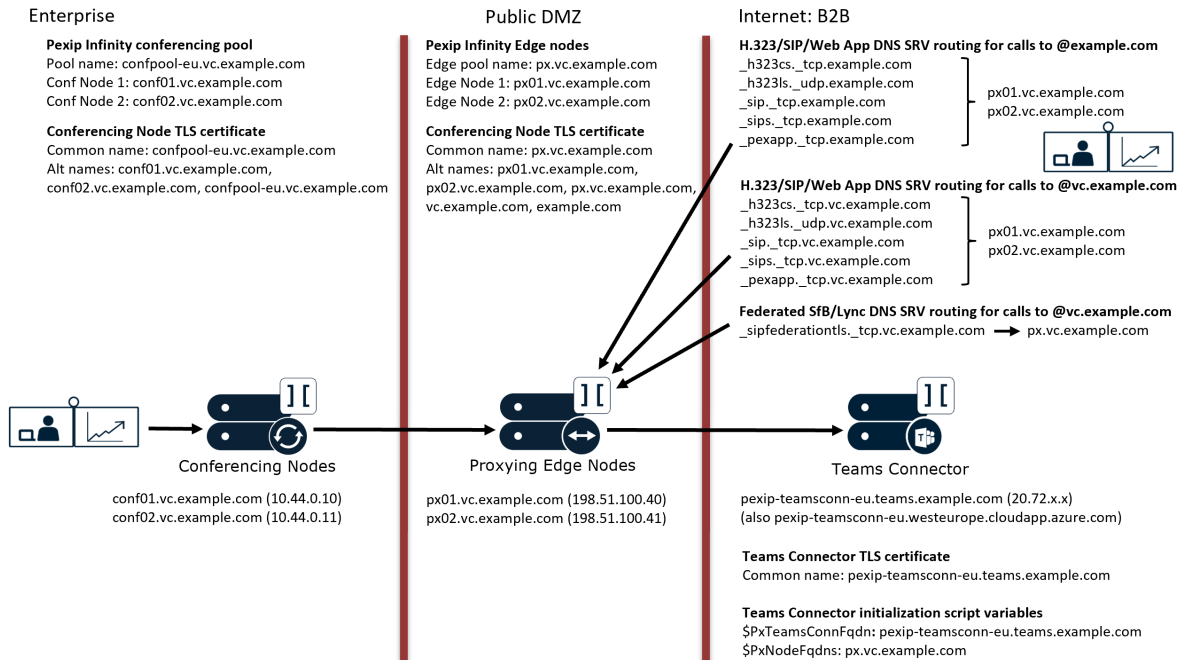
Join Microsoft Teams Meeting

[Learn more about Teams](#)

Join with a video conferencing device

teams@example.com VTC Conference ID: 123958530

[Alternate VTC dialing instructions](#)

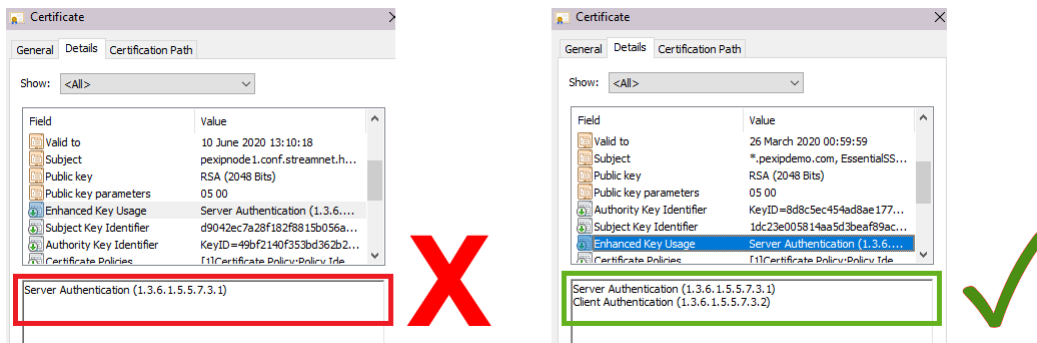


Note that the same Conferencing Nodes can be used for all other Pexip Infinity services, such as routing calls into Microsoft Skype for Business and Lync environments, and the certificate and DNS naming requirements and examples are compatible with all Pexip Infinity deployment scenarios.

Teams Connector and Conferencing Node certificates

In summary, the certificate usage principles are:

- The Teams Connector and Pexip Infinity validate the connection in both directions by TLS client certificate validation. This means that every certificate's Enhanced Key Usage properties must be set for both server **and** client authentication.



- Public-facing Conferencing Nodes must have a valid publicly-signed PEM-formatted certificate (typically with a .CRT or .PEM extension).
- The Teams Connector must have a publicly-signed PFX-formatted certificate. Multiple names/certificates are required if deploying Teams Connectors in several regions.

Teams Connector certificate requirements

You must install on the Teams Connector a TLS certificate that has been signed by an external trusted CA (certificate authority).

i You need to have this certificate available before you install the Teams Connector.

The certificate must be in Personal Information Exchange Format (PFX), also known as PKCS #12, which enables the transfer of certificates and their private keys from one system to another. It must use RSA keys.

1. Decide on the FQDN (DNS name) you will use for the Teams Connector load balancer in Azure that will front the Teams Connector deployment e.g. `pexip-teamsconn-eu.teams.example.com`.
 - This FQDN is what you will use as:
 - the value of `$PxTeamsConnFqdn` in the [variables initialization script](#)
 - the certificate's subject name
 - the DNS name you will configure in Pexip Infinity (Call Control > Microsoft Teams Connectors > Address Of Teams Connector) later in the process.
 - It can use the same domain space as your Pexip Infinity deployment (i.e. `vc.example.com` in this example), or your Teams deployment, or it can use an altogether different domain. In all cases you always need to create the necessary DNS CNAME record(s) and public certificates for the chosen domain.
 - If you intend to deploy other Teams Connectors in other Azure regions, you will need a different DNS name for each Teams Connector and a certificate that matches that identity. You can use a single certificate for this, containing Subject Alternative Name (altNames attribute) entries for all of the regional Teams Connectors.
 - It can be a wildcard certificate, where the wildcard character ('*') is the only character of the left-most label of a DNS domain name. Note that Pexip supports [RFC 6125](#) — this means that if you are using subdomains then, for example, a wildcard certificate of `*.example.com` would match `foo.example.com` but not `bar.foo.example.com` or `example.com`.
 2. Request a certificate for that name and generate the certificate in PFX format. Any intermediate certificates must also be in the PFX file.
- i** You can use the Pexip Infinity Management Node to generate a certificate signing request (CSR).
- i** You can use the Pexip Infinity Management Node to convert PEM certificates to PFX format (or vice versa), by uploading a PEM-formatted certificate and then downloading it again in PFX format. When downloading you can also include the private key and all necessary intermediate certificates in the PFX bundle.

Optional: internal/enterprise-based Conferencing Node guidelines

This section describes what is required for any internal/enterprise-based Conferencing Nodes. Enterprise-based nodes are optional and when used they can route calls from your internal VTC systems into Microsoft Teams conferences via the public DMZ-based Conferencing Nodes.

Certificates

For all of your enterprise-based Conferencing Nodes that are involved in call signaling with any on-premises VTC systems:

- We recommend that you generate and use a single SAN certificate that encompasses all of the enterprise-based Conferencing Nodes:
 - The Subject name should be a common pool name, such as `confpool-eu.vc.example.com` in our examples.
 - The Subject Alternative Name (altNames attribute) entries must include the Subject name plus the FQDNs of all of the nodes in the pool that are involved in any call signaling, such as `conf01.vc.example.com` and `conf02.vc.example.com` in our examples.
- Assign the same certificate to all of the enterprise Conferencing Nodes that are involved in call signaling.
- Conferencing Nodes require PEM-formatted certificates (typically with a `.CRT` or `.PEM` extension).

DNS records

In DNS, ensure that the following records are configured:

- An A-record for each Conferencing Node. In our example this is 2 records with host names of **conf01** and **conf02**, and they each point to the individual IP address of the node.

For example (change the hostnames and IP addresses as appropriate for your actual deployment):

```
conf01.vc.example.com.      86400 IN A 10.44.0.10
conf02.vc.example.com.      86400 IN A 10.44.0.11
```

To allow enterprise-based VTC systems to dial directly into Microsoft Teams meetings, or dial a Pexip Virtual Reception using aliases in the format `<alias>@example.com` (for example `teams@example.com`), you must configure local DNS SRV records. These example records will direct calls from H.323 devices, SIP devices and Connect apps (via the `_pexapp` SRV record) that are placed to the top-level `example.com` domain to your Conferencing Nodes in your private enterprise network (that are hosted in the `vc.example.com` subdomain):


```
_h323cs._tcp.example.com. 86400 IN SRV 10 10 1720 conf01.vc.example.com.
_h323cs._tcp.example.com. 86400 IN SRV 10 10 1720 conf02.vc.example.com.

_h323ls._udp.example.com. 86400 IN SRV 10 10 1719 conf01.vc.example.com.
_h323ls._udp.example.com. 86400 IN SRV 10 10 1719 conf02.vc.example.com.

_sip._tcp.example.com.      86400 IN SRV 10 10 5060 conf01.vc.example.com.
_sip._tcp.example.com.      86400 IN SRV 10 10 5060 conf02.vc.example.com.

_sips._tcp.example.com.     86400 IN SRV 10 10 5061 conf01.vc.example.com.
_sips._tcp.example.com.     86400 IN SRV 10 10 5061 conf02.vc.example.com.

_pexapp._tcp.example.com.   86400 IN SRV 10 10 443  conf01.vc.example.com.
_pexapp._tcp.example.com.   86400 IN SRV 10 10 443  conf02.vc.example.com.
```

In addition, those VTC devices and Connect apps may also want to call into Pexip Infinity services with addresses in the form **alias@vc.example.com**. The local DNS records to support those calls would be:

```
_h323cs._tcp.vc.example.com. 86400 IN SRV 10 10 1720 conf01.vc.example.com.
_h323cs._tcp.vc.example.com. 86400 IN SRV 10 10 1720 conf02.vc.example.com.

_h323ls._udp.vc.example.com. 86400 IN SRV 10 10 1719 conf01.vc.example.com.
_h323ls._udp.vc.example.com. 86400 IN SRV 10 10 1719 conf02.vc.example.com.

_sip._tcp.vc.example.com.     86400 IN SRV 10 10 5060 conf01.vc.example.com.
_sip._tcp.vc.example.com.     86400 IN SRV 10 10 5060 conf02.vc.example.com.

_sips._tcp.vc.example.com.    86400 IN SRV 10 10 5061 conf01.vc.example.com.
_sips._tcp.vc.example.com.    86400 IN SRV 10 10 5061 conf02.vc.example.com.

_pexapp._tcp.vc.example.com.  86400 IN SRV 10 10 443  conf01.vc.example.com.
_pexapp._tcp.vc.example.com.  86400 IN SRV 10 10 443  conf02.vc.example.com.
```

Public DMZ / Edge Conferencing Node guidelines

This section describes what is required for all of your public DMZ-based Conferencing Nodes that are involved in routing calls from your B2B and federated VTC systems into Microsoft Teams conferences. These examples assume that:

- The Pexip Infinity platform is on the **vc.example.com** VTC subdomain.
- There are two Pexip Conferencing Nodes in the public DMZ (**px01** and **px02**, and they have a DNS poolname of **px.vc.example.com**).

Your standard DNS A records for your public DMZ Conferencing Nodes will typically already exist:

```
px01.vc.example.com.      86400 IN A 198.51.100.40
px02.vc.example.com.      86400 IN A 198.51.100.41
```

Certificates

The Conferencing Nodes (typically Proxying Edge Nodes) that will communicate with the Teams Connector must have TLS certificates installed that have been signed by an external trusted CA (certificate authority). If a chain of intermediate CA certificates is installed on the Management Node (to provide the chain of trust for the Conferencing Node's certificate) those intermediate certificates must not include any HTTP-to-HTTPS redirects in their AIA (Authority Information Access) section.

The certificate guidelines are the same as for any internal Conferencing Nodes as described above, except that you should use a **different** pool name and certificate:

- We recommend that you generate and use a single SAN certificate that encompasses all of the public DMZ-based "Edge" Conferencing Nodes:
 - The Subject name should be a common pool name, such as **px.vc.example.com** in our examples.
 - The Subject Alternative Name (altNames attribute) entries must include the Subject name plus the FQDNs of all of the nodes in the pool that are involved in any call signaling, such as **px01.vc.example.com** and **px02.vc.example.com** in our examples.
- Assign the same certificate to all of the public DMZ-based Conferencing Nodes that are involved in call signaling and communications with the Teams Connector.
- Conferencing Nodes require PEM-formatted certificates (typically with a .CRT or .PEM extension).

DNS records for routing calls to @example.com

To allow external VTC systems to dial directly into Microsoft Teams meetings, or dial a Pexip Virtual Reception (for example **teams@example.com**) you must configure additional public DNS SRV records. These example records will direct calls from H.323 devices, SIP devices and Connect apps (via the **_pexapp** SRV record) that are placed to the top-level **example.com** domain to your Pexip Conferencing Nodes in the public DMZ (that are hosted in the **vc.example.com** subdomain):

```
_h323cs._tcp.example.com. 86400 IN SRV 10 10 1720 px01.vc.example.com.
_h323cs._tcp.example.com. 86400 IN SRV 10 10 1720 px02.vc.example.com.

_h323ls._udp.example.com. 86400 IN SRV 10 10 1719 px01.vc.example.com.
_h323ls._udp.example.com. 86400 IN SRV 10 10 1719 px02.vc.example.com.

_sip._tcp.example.com.      86400 IN SRV 10 10 5060 px01.vc.example.com.
_sip._tcp.example.com.      86400 IN SRV 10 10 5060 px02.vc.example.com.

_sips._tcp.example.com.     86400 IN SRV 10 10 5061 px01.vc.example.com.
_sips._tcp.example.com.     86400 IN SRV 10 10 5061 px02.vc.example.com.

_pexapp._tcp.example.com.   86400 IN SRV 10 100 443 px01.vc.example.com.
_pexapp._tcp.example.com.   86400 IN SRV 20 100 443 px02.vc.example.com.
```

Note that before configuring these DNS records you should check that there are no other **example.com** records for SIP and H.323 that are already configured and that are routing such calls elsewhere. (You can use the tool at <http://dns.pexip.com> to lookup and check SRV records for a domain.)

- i** You then need to configure appropriate Virtual Reception and Call Routing Rules on your Pexip Infinity system that will route those calls (placed to **@example.com**) onwards to Microsoft Teams. See [Configuring Pexip Infinity as a Microsoft Teams gateway](#) for more information.

DNS records for routing calls to @vc.example.com

In addition, the following public DNS SRV records will route calls from H.323 devices, SIP devices and Connect apps placed to your **@vc.example.com** subdomain to your Pexip Conferencing Nodes in the public DMZ:

```
_h323cs._tcp.vc.example.com. 86400 IN SRV 10 10 1720 px01.vc.example.com.
_h323cs._tcp.vc.example.com. 86400 IN SRV 10 10 1720 px02.vc.example.com.

_h323ls._udp.vc.example.com. 86400 IN SRV 10 10 1719 px01.vc.example.com.
_h323ls._udp.vc.example.com. 86400 IN SRV 10 10 1719 px02.vc.example.com.

_sip._tcp.vc.example.com.     86400 IN SRV 10 10 5060 px01.vc.example.com.
_sip._tcp.vc.example.com.     86400 IN SRV 10 10 5060 px02.vc.example.com.

_sips._tcp.vc.example.com.    86400 IN SRV 10 10 5061 px01.vc.example.com.
_sips._tcp.vc.example.com.    86400 IN SRV 10 10 5061 px02.vc.example.com.

_pexapp._tcp.vc.example.com.  86400 IN SRV 10 100 443 px01.vc.example.com.
_pexapp._tcp.vc.example.com.  86400 IN SRV 20 100 443 px02.vc.example.com.
```

Similarly you will then need suitable Call Routing Rules on your Pexip Infinity system to handle calls placed to **@example.com**.

Support for federated Skype for Business / Lync systems via the vc.example.com subdomain

If you want to enable federation for SfB/Lync clients to allow them to connect to Microsoft Teams (or other Pexip Infinity services) through your Pexip Conferencing Nodes in the public DMZ, you need a federation DNS SRV record for your Pexip Infinity subdomain that will handle calls placed to aliases in the format **alias@vc.example.com**.

Note that SfB/Lync clients should use direct routing aliases into Teams conferences. If they are routed via a Virtual Reception the call will revert to audio-only during the transfer.

The **_sipfederationtls._tcp.vc.example.com** DNS SRV and associated round-robin A-records shown below will route calls from federated SfB/Lync clients that are placed to the Pexip Infinity subdomain (**@vc.example.com**) to your Pexip Conferencing Nodes in the public DMZ (using the pool hostname **px.vc.example.com**):

```
_sipfederationtls._tcp.vc.example.com. 86400 IN SRV 1 100 5061 px.vc.example.com.
px.vc.example.com.                      86400 IN A 198.51.100.40
px.vc.example.com.                      86400 IN A 198.51.100.41
```

Note that these A-records specified for the **px.vc.example.com** pool are required in addition to the "standard" A-records that will exist for each Conferencing Node based on their individual hostnames and resolve to the same IP addresses.

More stringent configuration and certificate guidelines also apply when handling SfB/Lync calls:

- The SIP TLS FQDN setting on each Conferencing Node **must** match the node's DNS FQDN and it must be unique per node. For example, if the node's DNS FQDN is `px01.vc.example.com` then its SIP TLS FQDN setting must also be `px01.vc.example.com`.
- The certificate on each Conferencing Node **must** include the hostname referenced by the `_sipfederationtls._tcp` SRV record that points to those nodes, plus the names of all of the Conferencing Nodes that are involved in call signaling:
 - The Subject name (commonName attribute) should be set to the target hostname referenced by the `_sipfederationtls._tcp` SRV record (the pool name of the Conferencing Nodes).
In our examples, if the DNS SRV record is:
`_sipfederationtls._tcp.vc.example.com. 86400 IN SRV 1 100 5061 px.vc.example.com.`
then the Subject name must be `px.vc.example.com`
 - The Subject Alternative Name (altNames attribute) entries must include:
 - the target hostname referenced in the Subject name
 - the FQDNs of all of the public DMZ nodes that are involved in call signaling
 - the domain names that are used in any DNS SRV records that route calls to those Conferencing Nodes (e.g. `vc.example.com` from the example `_sipfederationtls` SRV record above).
 - Assign the same certificate to all of the public DMZ nodes that are involved in call signaling.
- The domain name used in the `_sipfederationtls._tcp.<domain>` SRV record has to match the domain in the corresponding A-record. This is required due to the trust model for SfB/Lync federation. For example:
 - An SRV record such as `_sipfederationtls._tcp.vc.example.com` must have a corresponding A-record with the same domain, such as `px.vc.example.com`.
 - You cannot, for example, configure the `_sipfederationtls._tcp.vc.example.com` SRV record to point to `px.video.example.com` or `px01.otherdomain.com`.

Note that the `_sipfederationtls._tcp` SRV record is used to route incoming calls from SfB/Lync clients. If you do not need to support SfB/Lync calls then you do not need a `_sipfederationtls._tcp` SRV record.

Migrating from — or co-existing with — a Skype for Business environment

If you have an existing Pexip Infinity integration with a Skype for Business / Lync environment, your existing DNS records and certificates should not need to change. Our certificate and DNS examples for on-prem and public DMZ / hybrid SfB/Lync integrations are exactly the same as the examples used here for our Microsoft Teams integration.

You only have to ensure that your Teams Connector certificate also meets our stated [requirements](#).

Pexip Infinity works simultaneously with both Microsoft Teams and Skype for Business. This means that users can be enabled to use both platforms and they can be migrated from one platform to the other at your own pace. Interoperability into either platform is handled by the same single Pexip Infinity installation, and the same Conferencing Nodes.

Typically your main domain (such as `example.com`) is used by your on-premises or Office 365 SfB/Lync environment, and your main federation DNS SRV record for SfB/Lync clients will already exist, which for our `example.com` domain it would typically be:

```
_sipfederationtls._tcp.example.com. 86400 IN SRV 1 100 5061 sip.example.com.
```

By using a subdomain for your Pexip environment i.e. `<subdomain>.example.com`, such as `vc.example.com` in our example, you can avoid any conflicts with your SfB/Lync environment.

However, by creating the H.323/SIP/Web App DNS SRV routing records for calls to `@example.com` as described above, you can provide a dial-in lobby address of:

- `skype@example.com` for interoperability into Skype for Business meetings

and

- `teams@example.com` for interoperability into Teams meetings

and then in each case the user would be directed to the appropriate Pexip Virtual Reception and would enter the appropriate conference ID for the relevant Microsoft meeting platform.

Your SfB/Lync-oriented routing rules in Pexip Infinity would then direct the SfB/Lync calls to your SfB/Lync environment, and your Teams-oriented routing rules would direct the Teams calls into Microsoft Teams via the Teams Connector.

Installing and configuring the Teams Connector in Azure

- i** No changes should be made to any Pexip Teams Connector or the provided deployment scripts (other than as described within this documentation for installing and maintaining your deployment) unless directed to do so by Pexip support. This includes (but is not limited to) any changes to the operating system or the installation of any third-party code/applications. If you encounter any issues, please contact your Pexip authorized support representative.

Deployment environment

You can deploy Teams Connector in the standard Azure commercial cloud, or where necessary for specific compliance requirements, in the Azure US Government environment.

Note that you can deploy Teams Connector in Azure US Government in combination with a commercial Teams environment or a Teams GCC environment. But, for Teams GCC High environments, Teams Connector must be deployed in Azure US Government.

- i** The Teams Connector must be hosted in the Arizona or Texas Azure regions for GCC High. The Virginia Azure region does not support the hosting of the Teams Connector for GCC High.

Most of the deployment processes are the same for either Azure environment, but there are some extra steps and minor differences when deploying in a US Government environment. Where there are variances indicated within this documentation you should either follow the **standard deployment** instructions (for commercial Azure) or the **GCC High / Azure US Government Cloud deployment** instructions.

Prepare your Azure environment, certificates and other prerequisites



















Ensure that you have decided upon your deployment environment and have obtained the necessary TLS certificates for your Teams Connector and Conferencing Nodes.

See [Planning, prerequisites and firewall ports](#) for details.

Download the Teams Connector application software files

You must download the Pexip Teams Connector application files to the administrator PC.

1. Download the **Teams Connector ZIP** file (Pexip_Infinity_Connector_For_Microsoft_Teams_v33_<build>.zip) from the [Pexip download page](#).
Ensure that the Teams Connector version you download is the same version as your Pexip Infinity deployment (including minor/"dot" releases).
2. Extract the files to a folder on a local drive on your administrator PC.
3. Verify that the ZIP file is extracted and that you see the following files:

> 	adminconsentwebapp	Folder
> 	function_apps	Folder
> 	function_apps_templates	Folder
	create_vmss_deployment.ps1	Document
	create_vnet_deployment.ps1	Document
	HelperFunctions.psm1	Document
	PexTeamsCviApplication.psm1	Document
	TeamsConnectorDSCConfig.ps1	Document
	test_pfx_certificate.ps1	Document
	appzip_hashes.json	JSON File
	azuredeploy-ac.json	JSON File
	azuredeploy-eventshub.json	JSON File
	azuredeploy-kv.json	JSON File
	azuredeploy-sa.json	JSON File
	azuredeploy-vnet.json	JSON File
	azuredeploy.json	JSON File
	version.json	JSON File
	app.zip	ZIP archive

4. Add your [PFX certificate](#) (that also contains all of your intermediates) for the Teams Connector to this folder.

 **your_connector_certificate.pfx** Personal Information Exchange

Install the Teams Connector application into Azure

Installation of the Teams Connector application is performed through PowerShell ISE commands and scripts.

These steps summarize the installation process, when installing the Teams Connector for the first time (per Azure subscription):

1. Update the variables in the initialization script with the real values for your environment, and then run the initialization script.
2. Create the resource groups used by the Teams Connector.
3. Create the Azure AD application used to secure the Teams Connector API and save the App ID in the variables initialization script.
4. Deploy the Teams Connector by running the installation script.
5. Save the Pexip CVI App ID and password output from the installation script into the redeploy script.
6. Update DNS with the Teams Connector's name and IP address.
7. Authorize the Pexip CVI application to join Teams meetings.
8. Authorize the app to bypass the Teams lobby and configure dialing instructions.

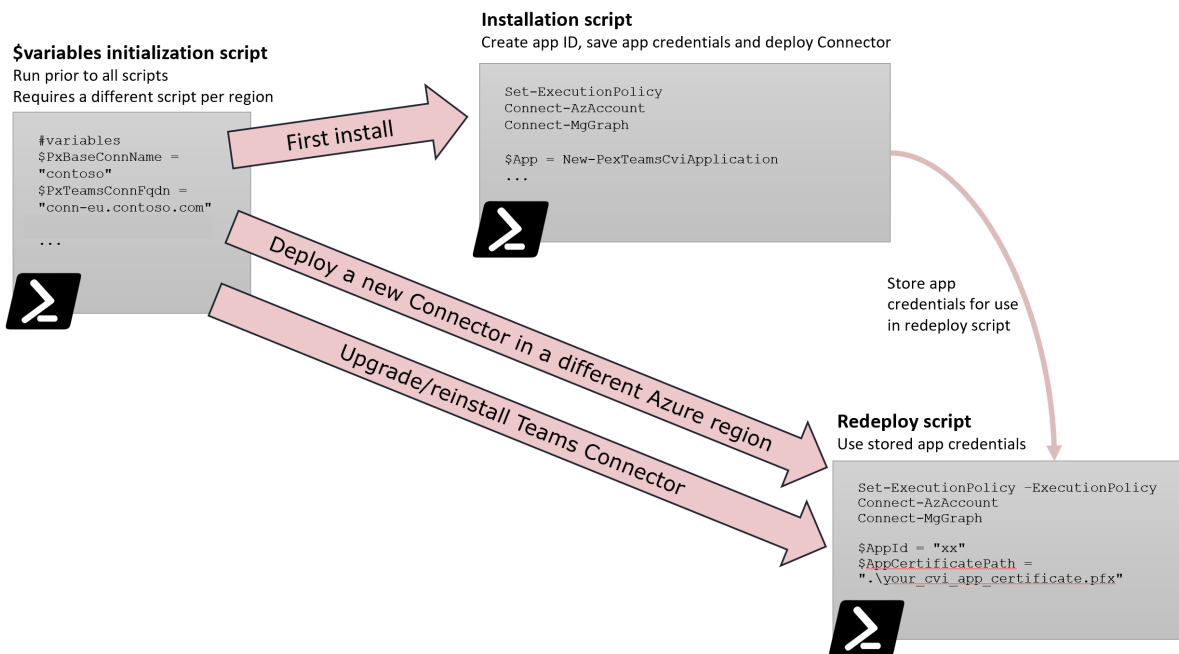
i The steps above describe the first-time installation process for a standard deployment. If you want to follow a blue-green deployment/upgrade strategy, see [Installing the Teams Connector using a blue-green deployment/upgrade strategy](#). If you subsequently need to redeploy or upgrade your Teams Connector, you need to follow the process described in [Maintaining your Teams Connector deployment](#). If you want to deploy another Teams Connector in a different Azure region, see [Installing Teams Connectors in multiple Azure regions](#).

From version 33 you can use certificate-based authentication (CBA) to authenticate the Teams Connector CVI application towards MS Graph. In version 33, CBA is optional and the previous password-based authentication method is still the default mechanism.

i The CBA method will be the default and recommended mechanism in version 34. Password-based authentication will still be supported in version 34 but we plan to deprecate it in a future release, thus we recommend migrating to CBA as soon as practicable.

If you want to start using CBA now, either when deploying Teams Connector for the first time, or when upgrading to version 33, see [Using certificate-based authentication for the Teams Connector CVI application](#).

The following diagram shows a summary of which scripts are used when initially installing and then subsequently maintaining your Teams Connector deployment. The variables initialization script is always the first script used in all scenarios.



General information about using PowerShell with Microsoft Graph and Office 365 can be found at <https://docs.microsoft.com/en-us/graph/powershell/get-started>.

Specify installation variables used in the PowerShell commands

You need to specify a range of PowerShell variables that are used during the installation process.

We recommend that you save these variables as a separate initialization script that you can reuse (and subsequently update if necessary). This will make it easier if you have to abort and restart the installation for any reason, or if you redeploy or upgrade your Teams Connector in the future.

- i** You must replace the example values set in the variables with the real values for your deployment.
- i** If you are deploying a Teams Connector in multiple regions, create a separate initialization script for each region, changing the relevant region-specific variables as appropriate. Save each region-specific version of your script in a safe place.

The PowerShell variables initialization script is listed below.

Note that this script does not produce any output. It only sets some variables for subsequent use in the installation script.

```
# Powershell variables
# Set the following variables for ease of use of the subsequent commands - if starting a new window, just re-set these variables.

# Name prefix for all Teams Connector resources, e.g. company name.
# PxBaseConnName can have a maximum of 9 characters
# PxBaseConnName + PxVmssRegion must be minimum 3 and maximum 14 chars when combined
# It has to begin with a letter and cannot contain dashes, spaces or other non a-z0-9 chars.
$PxBaseConnName = "yourname" # replace with your company name

# Freetext region shortname (2-4 characters, no dashes, only use a-z) to separate regional deployments and/or
# to differentiate between deployment versions if you are following a blue-green deployment strategy
$PxVmssRegion = "eu"

# Hostname of Teams Connector in Azure - Must match name in pfx certificate below
# You need a different hostname for each region
$PxTeamsConnFqdn = "pexip-teamsconn-eu.teams.example.com"
```

```

# Conference or Edge node pool (must be reachable from Teams Connector in Azure)
# This name must exist in the certificate presented by the Pexip nodes
# Example 1) Multiple individual Edge nodes
# $PxNodeFqdns = "us-pxedge01.vc.example.com,us-pxedge01.vc.example.com"
# Example 2) Certificate with SAN names, this name is in the cert presented by all nodes
$PxNodeFqdns = "pxedge.vc.example.com"

# Azure Subscription ID for Pexip Teams Connector deployment (GUID)
$PxSubscriptionId = "aaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"

# Azure Region (must be a supported region) in lower case
$PxAzureLocation = "westeurope"

# Username for the Windows VM accounts. Specifies the name of the administrator account.
# Windows-only restriction: Cannot end in "."
# Disallowed values: "administrator", "admin", "user", "user1", "test", "user2", "test1", "user3", "admin1",
# "123", "a", "actuser", "adm", "admin2", "aspnet", "backup", "console", "david", "guest", "john", "owner",
# "root", "server", "sql", "support", "support_388945a0", "sys", "test2", "test3", "user4", "user5", "1".
# It must be between 1 and 20 characters long
$PxWinAdminUser = "pexadmin"

# Password for the Windows VM administrator account
# Skip this step (setting the password) if you want to specify the password
# later from within the installation script (via Get-Credential)
# The password must include at least 3 of the following: 1 lower case character,
# 1 upper case character, 1 number, 1 special character that is not "\" or "-" or "$"
# It must be between 8 and 123 characters
# It must not include unsupported characters or reserved words (including "abc@123", "P@$w0rd", "P@ssw0rd",
# "P@ssword123", "Pa$$word", "pass@word1", "Password!", "Password1", "Password22", "iloveyou!")
$PxWinAdminPassword = "ReplacethisPassword!" # Password for Windows account

# Number of Teams Connector VMs (string "1"-100)
$PxTeamsConnInstanceCount = "3"

# Setting the dynamic regional resource group name
$PxTeamsConnResourceGroupName = "$($PxBaseConnName)-TeamsConn-$(PxVmssRegion)-RG"

# Setting the static regional resource group name
$PxTeamsConnStaticResourceGroupName = "$($PxBaseConnName)-TeamsConn-$(PxVmssRegion)-static-RG"

# Setting the Azure bot resource group name
$PxBotResourceGroupName = "$($PxBaseConnName)-TeamsBot-RG"

# Enable incident reporting
$PxTeamsConnIncidentReporting = $true

# Wildcard, SAN or single name cert for FQDN of Teams Connector (PxTeamsConnServiceFQDN),
# the PFX must contain the intermediate chain as well.
$PxPfxCertFileName = ".\your_connector_certificate.pfx"

# Public-facing IP address of management networks - used for RDP access
# If not specified (default) - RDP is always blocked
# Any security scans should not come from these IPs
# Example:
# x.x.x.x - Management IP address #1
# y.y.y.y - Management IP address #2
# z.z.z.z/24 - Management subnet
# $PxMgmtSrcAddrPrefixes = @( "x.x.x.x", "y.y.y.y", "z.z.z.z/24" )
$PxMgmtSrcAddrPrefixes = @()

# Pexip public facing Conferencing / Edge node IP addresses
# If not specified (default) - HTTPS access is always enabled
#
# Example Pexip Edge nodes public IP source ports:
# a.a.a.a - IP of us-pxedge01.vc.example.com
# c.c.c.c/28 - IP subnet of eu-pxedges (allows for future expansion)
#
# Example (specifying Pexip Edge nodes and Management networks defined above):
# $PxNodesSourceAddressPrefixes = @( "a.a.a.a", "c.c.c.c/28" ) + $PxMgmtSrcAddrPrefixes

```



```

$PxNodesSourceAddressPrefixes = @()

# These are the IPs/subnets that are allowed to access the Admin Consent web page.
# If not specified it is exposed/public by default. If left exposed then anyone accessing this web page can
# see the company's app IDs. You can alternatively stop the app service after granting consent.
# The page URL is in the form: https://$PxBaseConnName-pexip-cvi-abcde.azurewebsites.net/
#
# Example (specifying the Pexip management networks defined above):
# $PxConsentSourceAddressPrefixes = $PxMgmtSrcAddrPrefixes
$PxConsentSourceAddressPrefixes = @()

# Schedule installation of Windows updates to a specific day.
# String in range "0"-"7".
# "0" = Every day. "1" through "7" = The days of the week from Sunday ("1") to Saturday ("7")
$PxWupdScheduledInstallDay = "1"

# Schedule update installation time to a specific hour (UTC)
# String in range = "0"-"23" starts with 12 AM ("0") and ends with 11 PM ("23")
$PxWupdScheduledInstallTime = "22"

# Set active hours to start at a specific hour (UTC)
# If the restart is needed to finish update installation, it won't take place during the active hours.
# String in range = "0"-"23" starts with 12 AM ("0") and ends with 11 PM ("23")
$PxWupdActiveHoursStart = "6"

# Set active hours to end at a specific hour (UTC)
# Time between Active hours start and end is limited to 18 hours. Script stops if this is not the case.
# String in range = "0"-"23" starts with 12 AM ("0") and ends with 11 PM ("23")
$PxWupdActiveHoursEnd = "22"

# Do you want to allow Microsoft to track and report to Pexip the Azure usage associated with this deployment?
# You must set $PxCustomerUsageAttribution to either $true (allow reporting) or $false (no reporting)
$PxCustomerUsageAttribution = <replace with $true or $false>

# Optional tags (name-value pairs) to apply to Azure resources and resource groups
# For example $tags= @{ "ResourceOwner"="user@domain"; "CostCenter"="Video Services"; }
$tags= @{}

# Teams Connector API app ID - update this with the API App ID that is created during the installation process
# If you are deploying in multiple regions you will have a different App ID per region
$TeamsConnectorApiApplicationId = ""

# If set to $true, the Azure functions apps are deployed using dedicated function app plan.
# This incurs additional charges, but allows for advanced networking configuration and increased stability.
$FunctionsDedicatedHostingPlan = $false

# IP addresses of the Pexip Management Node / NAT gateway that can communicate with the Teams Connector Event
# Hub over port 5671.
# This only applies if used together with dedicated functions and VNet integration:
# (parameter FunctionsDedicatedHostingPlan and parameter VnetIntegration).
# These addresses can also be added after the deployment.
# Example: Pexip Management Node public IP address or NAT address:
# x.x.x.x - IP of mgr.vc.example.com
# z.z.z.0/24 - IP subnet of mgr.vc.example.com
$EventHubSourceAddressPrefixes = @()

# If set to $true, certain resources (key vault, storage accounts and event hubs) will be
# accessible via virtual network instead of public endpoint.
# (With the exception of the Event hub -> Management Node connection.)
# This only applies if used together with dedicated functions (parameter FunctionsDedicatedHostingPlan).
$VnetIntegration = $false

# Set to $true if your organization participates in Microsoft's Azure Hybrid Benefit scheme
$PxUseAzureHybridBenefit = $false

# The FQDN of the Teams Connector from the perspective of Pexip Infinity - it must match the name in
# the Connector's PFX certificate.
# If the Teams Connector sits behind an additional proxy, this should be set to the FQDN of that proxy.
# If not set it defaults to TeamsConnectorFqdn ($PxTeamsConnFqdn), which is normally correct.
$PexipConfiguredConnectorFqdn = ""

```

```
# This is used for Microsoft Teams Rooms SIP/H.323 calling.
# Set it to the DNS name of the nodes to target when dialing out to Pexip Infinity from a Teams Room.
# It should represent all external nodes and could be either round-robin DNS, a reverse proxy, or
# anything else that load balances across Pexip Infinity.
$PexipOutboundFqdn = ""


# Optional setting. Contains the resource ID of an existing (pre-created) VNET.
# Referencing a VNET in a different subscription is not supported.
# You must use the script create_vnet_deployment.ps1 to create this VNET.
$PxExistingVNETResourceId = ""

# If set to $true, an additional internal load balancer is created (to support private routing).
# VNET peering must be enabled between the Infinity VNET and the Teams Connector's VNET.
# When Infinity is deployed in Azure, it allows traffic to be routed between Infinity and the
# Teams Connector through Microsoft's private network only.
$PxUsePrivateRouting = $false
```

- i** The initialization script contains the following variables. If you are deploying a Teams Connector in multiple regions, each version of your script per region should use a different value for those variables ticked as **Regional**.

Variable name	Description and example usage	Regional
\$PxBaseConnName	<p>This is a prefix used when naming all Teams Connector resources. We recommend using your own company name.</p> <p>PxBaseConnName can have a maximum of 9 characters, and PxBaseConnName + PxVmssRegion must be a minimum 3 and maximum 14 characters when combined. It cannot contain dashes, spaces or other non a-z0-9 characters, and must start with an alphabetic character.</p> <p>Note that if you are setting up multiple test environments within the same Azure subscription, ensure that each Teams Connector deployment has a unique \$PxBaseConnName.</p>	
\$PxVmssRegion	<p>A short (we recommend 2-8 characters) name to differentiate resource names in your deployments:</p> <ul style="list-style-type: none"> It can be used to represent the region in which you are deploying the Teams Connector, for example, "eu". This will help in your naming convention if you deploy the Teams Connector in multiple regions. You can also use this variable to name your deployments if you are following a blue-green deployment strategy, for example "blue". If you have multiple regions and are also following a blue-green strategy then just combine the two names, for example "eublu". <p>It cannot contain dashes, spaces or other non a-z0-9 characters.</p>	✓
\$PxTeamsConnFqdn	<p>The hostname of the Teams Connector, for example "pexip-teamsconn-eu.teams.example.com".</p> <p>This name must match the subject name in the PFX certificate specified in \$PxPfxCertFileName. If you are installing multiple Teams Connectors in different regions you must use a different hostname for each region.</p> <p>This is also the name you will configure in Pexip Infinity (Call Control > Microsoft Teams Connectors > Address Of Teams Connector) later in the process.</p>	✓

Variable name	Description and example usage	Regional
\$PxNodeFqdns	<p>The pool name or names of the Conferencing Nodes (typically Proxying Edge Nodes) that will communicate with the Teams Connector. This can be a comma-separated list.</p> <p>The name(s) specified here must exist in the certificate presented by those Conferencing Nodes. These names cannot include wildcard characters.</p> <p>See Ensuring Conferencing Nodes have suitable certificates for more information.</p>	✓ (typically)
\$PxSubscriptionId	The Azure Subscription ID for your Teams Connector installation and Azure bot resource. This takes the GUID format "aaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee".	
\$PxAzureLocation	The name of the Azure region into which you are deploying the Teams Connector, for example "westeurope", "southcentralus" etc. as per the RegionName in Decide Azure deployment region(s) and check quota .	✓
\$PxWinAdminUser	<p>The account username for the Windows VMs that will be created in Azure. You may need this for RDP login when troubleshooting.</p> <p>See this Microsoft article for username requirements.</p>	
\$PxWinAdminPassword	<p>The account password for the Windows Teams Connector VMs that will be created in Azure. If you don't want to store the password in the variables script, you can skip this step and specify the password later from within the installation script (via Get-Credential cmdlet).</p> <p>The password must:</p> <ul style="list-style-type: none"> include at least 3 of the following: <ul style="list-style-type: none"> 1 lower case character 1 upper case character 1 number 1 special character that is not "\" or "-" or "\$" be between 8 and 123 characters long (must have a minimum of 14 characters in GCC High / Azure US Government Cloud deployments) not include reserved words or unsupported characters. 	
\$PxTeamsConnInstanceCount	<p>The number of Teams Connector instances (VMs) to deploy.</p> <p>A string in the range "1"-100", for example "3".</p> <p>Each instance can handle approximately 10 VTC connections into Teams meetings.</p> <p>You can easily modify the number of instances via the Azure portal after installation of the Teams Connector, to reflect changing capacity requirements.</p>	
\$PxTeamsConnResourceGroupName	<p>The regional resource group name for dynamic VMSS resources. We recommend setting this variable to a concatenated combination of the first two variables plus some additional text, for example:</p> <pre>\$PxTeamsConnResourceGroupName = "\${PxBaseConnName}-TeamsConn-\${PxVmssRegion}-RG"</pre> <p>which, in our examples, would generate a resource group name of pexip-TeamsConn-eu-RG.</p>	

Variable name	Description and example usage	Regional
\$PxTeamsConnStaticResourceGroupName	<p>The static regional resource group name. This stores static resources such as name and address information for the load balancer, and the Azure Key Vault. It should follow the same pattern as the previous variable (\$PxTeamsConnResourceGroupName), and be set to a concatenated combination of the first two variables plus some additional text, for example:</p> <pre>\$PxTeamsConnStaticResourceGroupName = "\${PxBaseConnName}-TeamsConn-\${PxVmssRegion}-static-RG"</pre> <p>which, in our examples, would generate a static resource group name of pexip-TeamsConn-eu-static-RG.</p> <p> This is the resource group that must be created before running the installation script (see Create the resource groups).</p>	
\$PxBotResourceGroupName	<p>The Azure bot resource group name. It should follow the same pattern as the other resource group names. We recommend:</p> <pre>\$PxBotResourceGroupName = "\${PxBaseConnName}-TeamsBot-RG"</pre>	
\$PxTeamsConnIncidentReporting	<p>When this feature is enabled, incident reports are sent automatically to a secure web server owned and managed by Pexip. The options are \$true to enable incident reporting or \$false to disable reporting. We recommend keeping this enabled.</p>	
\$PxPfxCertFileName	<p>The filename of the PFX certificate file to upload to the Teams Connector. See Obtaining and preparing the TLS certificate for the Teams Connector for more information.</p>	✓
\$PxMgmtSrcAddrPrefixes	<p>Specifies the public-facing IP addresses of any management workstations/networks that may be required to administer the Teams Connector instances over RDP.</p> <p>Any addresses specified here are added into the Azure Network Security Group "RDP" rule that is assigned to the Teams Connector instances to allow RDP access to those instances from those addresses. You should not perform any security scans from these addresses.</p> <p>For example to allow RDP access from:</p> <pre>x.x.x.x Management IP address #1 y.y.y.y Management IP address #2 z.z.z.0/24 Management subnet</pre> <p>you would specify \$PxMgmtSrcAddrPrefixes = @("x.x.x.x", "y.y.y.y", "z.z.z.0/24")</p> <p>If no addresses are specified i.e. \$PxMgmtSrcAddrPrefixes = @() then all RDP access is blocked and can only be enabled in the future by specifying a value for this variable and redeploying.</p>	

Variable name	Description and example usage	Regional
\$PxNodesSourceAddressPrefixes	<p>Specifies the public IP addresses of the Conferencing Nodes (typically Proxying Edge Nodes) that can communicate with the Teams Connector instances over port 443 (https).</p> <p>This is used to populate the Azure Network Security Group "MCU-signalling-endpoint" rule.</p> <p>For example, if these are the public addresses of your Pexip Conferencing Nodes:</p> <pre>a.a.a.a - IP of us-pxedge01.vc.example.com c.c.c.0/28 - IP subnet of eu-pxedges (allows for future expansion)</pre> <p>you would specify:</p> <pre>\$PxNodesSourceAddressPrefixes = @("a.a.a.a", "c.c.c.0/28") + \$PxMgmtSrcAddrPrefixes</pre> <p>which sets the variable to the addresses of your Conferencing Nodes plus the IP addresses of any management workstations/networks (as specified in the previous variable).</p> <p>If no addresses are specified i.e. <code>\$PxNodesSourceAddressPrefixes = @()</code> then anything can connect via https to the Teams Connector instances.</p>	✓ (typically)
\$PxConsentSourceAddressPrefixes	<p>Specifies the external IP address (as seen from the Teams Connector's perspective) of the workstation / management network that will provide consent for the Pexip CVI app to access Microsoft Teams in the Office 365 tenant.</p> <p>If not specified, the consent page is exposed/public by default. If left exposed then anyone accessing this page can see the company's app IDs. As an alternative to restricting access, you can stop the Azure app service after granting consent (the app service is in the <prefix>-TeamsBot-RG resource group).</p> <p>The URL of the consent page is in the form: <code>https://<prefix>-pexip-cvi-abcde.azurewebsites.net/</code>.</p> <p>For example, to use the Pexip management networks defined above you would specify:</p> <pre>\$PxConsentSourceAddressPrefixes = \$PxMgmtSrcAddrPrefixes</pre> <p>Alternatively, you can specify specific addresses in the same way as shown above when defining the values for <code>\$PxNodesSourceAddressPrefixes</code>.</p> <p>If no addresses are specified i.e. <code>\$PxConsentSourceAddressPrefixes = @()</code> then anything can connect via https to the consent page.</p>	
\$PxWupdScheduledInstallDay	<p>Schedules the installation of Windows updates (everything that applies to Windows Server 2016) to a specific day.</p> <p>A string in the range "0"-"7" where "0" = every day and "1" through "7" are the days of the week from Sunday ("1") to Saturday ("7").</p> <p>Default = "1" (Sunday)</p> <p>Example: <code>\$PxWupdScheduledInstallDay = "1"</code></p>	✓

Variable name	Description and example usage	Regional
\$PxWupdScheduledInstallTime	<p>Schedules the Windows update installation time to a specific hour (UTC).</p> <p>A string in the range "0"-"23" starting with 12 AM ("0") and ending with 11 PM ("23").</p> <p>Default = "22" (10 PM)</p> <p>Example: <code>\$PxWupdScheduledInstallTime = "22"</code></p>	✓
\$PxWupdActiveHoursStart	<p>Sets the active (business) hours to start at a specific hour (UTC). If a restart is needed to finish a Windows update, it won't take place during the active hours.</p> <p>A string in the range "0"-"23" starting with 12 AM ("0") and ending with 11 PM ("23").</p> <p>Default = "6" (6 AM)</p> <p>Example: <code>\$PxWupdActiveHoursStart = "6"</code></p>	✓
\$PxWupdActiveHoursEnd	<p>Sets active (business) hours to end at a specific hour (UTC). The time between Active hours start and Active hours end is limited to a maximum of 18 hours. The script stops if this is not the case.</p> <p>A string in the range "0"-"23" starting with 12 AM ("0") and ending with 11 PM ("23").</p> <p>Default = "22" (10 PM)</p> <p>Example: <code>\$PxWupdActiveHoursEnd = "22"</code></p>	✓
\$PxCustomerUsageAttribution	<p>Controls whether Microsoft tracks and reports to Pexip the Azure usage that is associated with your deployment of Pexip software. When enabled, Microsoft can identify the installation of Pexip software and correlate the Azure resources that are used to support that software. Microsoft collects this information to provide the best experiences with their products and to operate their business. The data is collected and governed by Microsoft's privacy policies, which can be found at https://www.microsoft.com/trustcenter.</p> <p>The data made visible to Pexip is controlled by Microsoft, and as of April 2019 includes usage and spend on all of the Azure resources associated with your Teams Connector.</p> <p>You can also track your usage of resources within the Azure portal.</p> <p>To enable usage tracking and reporting to Pexip, use:</p> <pre>\$PxCustomerUsageAttribution = \$true</pre> <p>To disable usage tracking and reporting to Pexip, use:</p> <pre>\$PxCustomerUsageAttribution = \$false</pre> <p>You must set the <code>\$PxCustomerUsageAttribution</code> variable to either <code>\$true</code> or <code>\$false</code> otherwise the installation will not complete.</p>	
\$tags	<p>You can optionally specify a set of tags (name-value pairs) to apply to the Azure resources and resource groups that are created for the Teams Connector. For example, to apply a tag named "ResourceOwner" with a value of "user@domain" and a second tag named "CostCenter" with a value of "Video Services" you would use:</p> <pre>\$tags= @{"ResourceOwner"="user@domain"; "CostCenter"="Video Services";}</pre> <p>To not apply any tags, leave the variable set to an empty hash table:</p> <pre>\$tags= @{}</pre>	

Variable name	Description and example usage	Regional
\$TeamsConnectorApiApplicationId	<p>The ID of the Azure AD application that is used to secure requests to the Teams Connector APIs.</p> <p>It is created during the installation process as described below at Create the Teams Connector API app, and you should then set this variable to the ID generated by that process.</p> <p>If you are deploying in multiple regions you'll have a different API App ID per region.</p>	✓
\$FunctionsDedicatedHostingPlan	<p>If set to \$true, the Azure functions apps are deployed using a dedicated function app plan. This incurs additional charges, but allows for advanced networking configuration and increased stability.</p> <p>Set it to \$false if you do not want to use dedicated functions.</p>	
\$EventHubSourceAddressPrefixes	<p>Specifies the IP addresses of the Pexip Management Node / NAT gateway that can communicate with the Teams Connector Event Hub over port 5671.</p> <p>This only applies if used together with dedicated functions and VNet integration (parameter <code>FunctionsDedicatedHostingPlan</code> and parameter <code>VnetIntegration</code>).</p> <p>These addresses can also be added after the deployment.</p> <p>For example to allow access from:</p> <pre>x.x.x.x Management IP address #1 y.y.y.y Management IP address #2 z.z.z.0/24 Management subnet</pre> <p>you would specify <code>\$EventHubSourceAddressPrefixes = @("x.x.x.x", "y.y.y.y", "z.z.z.0/24")</code></p> <p>Specify <code>\$EventHubSourceAddressPrefixes = @()</code> if you are not using these features.</p>	
\$VnetIntegration	<p>If set to \$true, certain resources (Key Vault, storage accounts and Event Hubs) will be accessible via the virtual network instead of public endpoint (with the exception of the Event Hub to Management Node connection). It only applies if used together with dedicated functions (parameter <code>FunctionsDedicatedHostingPlan</code>).</p> <p>Set it to \$false if you do not want to use VNet integration.</p>	
\$PxUseAzureHybridBenefit	<p>Set this option to \$true if your organization uses Azure Hybrid Benefit and you want to take advantage of this licensing for your Teams Connector virtual machines. For more information, see Microsoft's website.</p> <p>By default this is \$false</p>	
\$PexipConfiguredConnectorFqdn	<p>This is used for Microsoft Teams Rooms SIP/H.323 calling, and is typically only required in advanced deployments.</p> <p>Set it to the FQDN of the Teams Connector from the perspective of Pexip Infinity — it must match the name in the Teams Connector's PFX certificate.</p> <p>If the Teams Connector sits behind an additional proxy, such as an Azure Traffic Manager, this should be set to the FQDN of that proxy.</p> <p>If not set it defaults to <code>TeamsConnectorFqdn (\$PxTeamsConnFqdn)</code>, which is normally suitable for most standard deployments.</p>	✓

Variable name	Description and example usage	Regional
\$PexipOutboundFqdn	<p>This is used for Microsoft Teams Rooms SIP/H.323 calling.</p> <p>Set it to the DNS name of the Conferencing Nodes to target when dialing out to Pexip Infinity from a Teams Room. It should represent all external nodes and could be either round-robin DNS, a reverse proxy, or anything else that load balances across Pexip Infinity.</p> <p>These nodes must be reachable from the Teams Connector in Azure.</p>	
\$PxExistingVNETResourceId	<p>This is an optional variable. It contains the resource ID of an existing (pre-created) VNET and is used for enabling private routing*.</p> <p>Default = ""</p> <p>See Using private routing with the Teams Connector for more information.</p>	
\$PxUsePrivateRouting	<p>If set to \$true, an additional internal load balancer is created (to support private routing)*.</p> <p>Default = \$false</p> <p>See Using private routing with the Teams Connector for more information.</p>	

Enabling the CIS STIG image (GCC High / Azure US Government Cloud deployments only)

Please skip this section if you are following a standard deployment model.

If this is a GCC High / Azure US Government Cloud deployment, and you are using the CIS STIG image, there are some additional steps required to enable the image in Azure.

Note that using the CIS STIG image is optional, but typical, in GCC High deployments. If you do not want to use a STIG image you can skip this section and also remove the VmlImage switch from the installation script.

1. Verify that the CIS image is available in your subscription:
 - Run the following command:


```
Get-AzVMImageSku -Location USGovTexas -PublisherName center-for-internet-security-inc -Offer cis-win-2019-stig
```
2. Accept terms for the STIG image:
 - Run the following commands:


```
$agreementTerms=Get-AzMarketplaceTerms -Publisher "center-for-internet-security-inc" -Product "cis-win-2019-stig" -Name "latest"
Set-AzMarketplaceTerms -Publisher "center-for-internet-security-inc" -Product "cis-win-2019-stig" -Name "cis-win-2019-stig" -Terms $agreementTerms -Accept
```
3. Configure Programmatic Deployment in Azure Marketplace for the STIG image:
 - a. In the Azure Marketplace, search for "CIS 2019 STIG".
 - b. Click **CIS Microsoft Windows Server 2019 STIG Benchmark**.
 - c. Click **Get started**.
 - d. Click **Enable** and then **Save**.

Installation commands to connect to Microsoft Graph, Azure Resource Manager and deploy the Teams Connector

This section describes the Azure permissions, steps and PowerShell commands used to install the Teams Connector.

First-time use

If you are connecting to Azure Resource Manager / Microsoft Graph from your Windows PC for the first time, you must run the following PowerShell commands (as Administrator):

```
Install-Module -Name Az -MinimumVersion 9.0.1 -MaximumVersion 9.7.1 -AllowClobber -Scope AllUsers
Install-Module Microsoft.Graph -MinimumVersion 1.28.0 -MaximumVersion 2.2.0 -AllowClobber -Scope AllUsers
```

Note that:

- The installation of Microsoft Graph PowerShell SDK can take 5-10 minutes. Wait until you get the PS prompt back (several minutes after the install reports as completed) before continuing.
- The Az PowerShell module collects telemetry data (usage data) by default, however you can opt out from this data collection using `Disable-AzDataCollection` (see [this article](#) for more information).
- The Az PowerShell module remembers login information by default (i.e. you're not automatically logged out when closing your PowerShell window), but you can disable this with `Disable-AzContextAutosave` if required (see [this article](#) for more information).

Azure permissions requirements


You (the person performing the installation) must have appropriate Azure permissions for some of the resources used for the Teams Connector:

- You must have the Azure **Owner** role for the static and dynamic resource groups, and **Contributor** role for the Azure Bot resource group that are created (as described below) for each Teams Connector, and the **Owner** role for the API app.
- If the Azure AD tenant is configured with **Users can register applications** set to **No**, then appropriate permissions are also required to register the Pexip apps. If the applications are created by another user, then the user that is running the deployment script needs to have been given owner permissions of the applications. (See [this article](#) for more information).
- We recommend that the Teams Connector is deployed by a member account within the tenant. However, if it is deployed by a guest user within the tenant, that guest user also requires the **Directory Readers** role for the tenant.
- The user account used for deployment does not need to retain any RBAC permissions to the Teams Connector resource groups. The roles assigned can be removed or downgraded after deployment if this is a local policy requirement. However, appropriate roles would have to be re-assigned to upgrade the Teams Connector.
- We recommend reviewing the role assignments that grant access to the deployed resources, with special attention to inherited RBAC roles.

Create the resource groups

Before you install the Teams Connector CVI application you must first create all of the resource groups that are to be used for each Teams Connector. This is a mix of static resource groups that persist whenever the Teams Connector is upgraded, and dynamic resource groups that need recreating whenever you upgrade or redeploy a Teams Connector.

To set up the resource groups:

 These steps **must** be performed by the **Owner** of the Azure subscription used for the Teams Connector.

1. Run the following PowerShell command to connect to Azure:
 - In all standard deployments run:
`Connect-AzAccount`
 - If this is a GCC High / Azure US Government Cloud deployment then use this command instead:
`Connect-AzAccount -EnvironmentName AzureUSGovernment`

Then follow the prompts to sign in to Azure.

2. Run the variable initialization script (to set the required subscription, resource group name and region variables).
3. Ensure that you are using the Azure subscription for the Teams Connector:
`Set-AzContext -SubscriptionId $PxSubscriptionId`
4. Run the following script to create the resource groups:

```
# Resource group for static resources for the Teams Connector / region
New-AzResourceGroup -Name $PxTeamsConnStaticResourceGroupName -Location $PxAzureLocation -Force -Tag $tags

# Resource group for the Teams Connector VMSS (per region)
New-AzResourceGroup -Name $PxTeamsConnResourceGroupName -Location $PxAzureLocation -Tag $tags

# Resource group for the Azure Bot
New-AzResourceGroup -Name $PxBotResourceGroupName -Location $PxAzureLocation -Tag $tags
```

- Ensure that the person who will perform all of the remaining installation steps has the Azure **Owner** role for the static and dynamic resource groups, and **Contributor** role for the Azure Bot resource group you have just created. If the Owner of the Azure subscription will perform all of the remaining installation steps then you can skip to [Create the Teams Connector API app](#) below. Otherwise, you must run the following commands to assign the required roles for the **resource groups** you have just created to the person who will perform all of the remaining installation steps:

```
New-AzRoleAssignment -SignInName <email_name> -RoleDefinitionName "Owner" -ResourceGroupName
$PxTeamsConnStaticResourceGroupName
New-AzRoleAssignment -SignInName <email_name> -RoleDefinitionName "Owner" -ResourceGroupName $PxTeamsConnResourceGroupName
New-AzRoleAssignment -SignInName <email_name> -RoleDefinitionName "Contributor" -ResourceGroupName $PxBotResourceGroupName
```

where **<email_name>** is the email address / user principal name of the person who will perform the remaining installation steps; for example where `alice@example.com` will perform the upgrade/redeploy, the `SignInName` would be `-SignInName alice@example.com` for the commands listed above.

Note:

- Some of these resource groups only need creating once; others need recreating when you upgrade or if you redeploy:
 - The static resource group (`$PxTeamsConnStaticResourceGroupName`) only has to be created when deploying in a region for the first time — you do not have to repeat this when redeploying or for subsequent upgrades.
 - The dynamic resource group (`$PxTeamsConnResourceGroupName`) has to be recreated in each region whenever you upgrade or redeploy.
 - The Azure Bot resource group (`$PxBotResourceGroupName`) only has to be created once, and only in your first region — you do not have to repeat this when redeploying or for subsequent upgrades.
- In the future, if another person were to upgrade or redeploy the Teams Connector, that person would also have to be granted the appropriate roles for these resource groups.
- You can also use the Azure portal to check/assign permissions by selecting the resource group and using the **Access control (IAM)** option.

Create the Teams Connector API app

You must create an Azure AD application that is used to secure requests to the Teams Connector APIs.

- Run the following PowerShell command to connect to Azure:
 - In all standard deployments run:


```
Connect-AzAccount
```
 - If this is a GCC High / Azure US Government Cloud deployment then use this command instead:


```
Connect-AzAccount -EnvironmentName AzureUSGovernment
```

Then follow the prompts to sign in to Azure.

- Change directory to the folder into which you extracted the files from the Teams Connector ZIP.
- Run the following PowerShell commands to connect to Microsoft Graph:

```
# Set execution policy for the current PowerShell process, when prompted type A (Yes to All)
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# The Unblock-File cmdlet lets you run PowerShell script files that were downloaded from the internet.
# By default, these files are blocked to protect the computer from untrusted files.
Get-Childitem -Recurse | Unblock-File

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph
```

- Run your variable initialization script to set the required prefix and region name variables.
- Run the following command to create the Teams Connector API app.

```
$teamsConnectorApiApp = New-MgApplication -DisplayName "${PxBaseConnName}-TeamsConn-${PxVmssRegion} Pexip Teams Connector API" -
SignInAudience "AzureADMyOrg"
```

Please allow one minute for this command to complete and propagate within Azure.

- Run the following commands to obtain the Teams Connector API app ID.

Note that if the **New-MgServicePrincipal** command fails, this means that you have not waited long enough for the previous command to complete.

```
$TeamsConnectorApiSp = New-MgServicePrincipal -AppId $teamsConnectorApiApp.AppId -Tags {WindowsAzureActiveDirectoryIntegratedApp}
$TeamsConnectorApiApplicationId = $teamsConnectorApiApp.AppId

Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host "### Teams Connector API App ID MUST be saved in the variables initialization script ###"
Write-Host
Write-Host "$TeamsConnectorApiApplicationId = `"$($TeamsConnectorApiApplicationId)`""
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host
```

- When the command runs, it generates some output that lists the Teams Connector API App ID, similar to this:

```
### Teams Connector API App ID MUST be saved in the variables initialization script ###
$TeamsConnectorApiApplicationId = "36ee4c6c-0812-40a2-b820-b22ebd02bce4"
```

- Copy the output line that defines the API App ID and paste it into the variable initialization script, replacing the existing line in the script that says:

```
$TeamsConnectorApiApplicationId = ""
```

(If you will be performing the rest of the deployment, in the same PowerShell session, there is no need to re-run the variable initialization script as the new \$TeamsConnectorApiApplicationId variable has been set in the previous steps.)

- If somebody else will be completing the deployment, ensure that the person who will perform all of the remaining installation steps has **Owner** permissions for the API app. See [Assigning Owner permissions for the API app](#) for more information.

Note that:

- This app does not have to be granted any permissions. It does not have access to any resources in the Azure AD tenant. It has no associated credentials.
- It is different from the main Pexip CVI App which is created by the installation script.
- If you intend to deploy the Teams Connector in multiple Azure regions, you must create an API app in each region, and store the app ID (\$TeamsConnectorApiApplicationId) in the relevant variable initialization script for that region.

Deploying the Teams Connector

After creating the resource groups and the Teams Connector API App, you can now use the following PowerShell script and commands to connect to Microsoft Graph, connect to Azure Resource Manager, set up the Pexip CVI App ID, and then deploy the Teams Connector. The purpose of each command is explained as a comment within the script.

- If you need to upgrade or redeploy your Teams Connector you should use the [redeploy process](#) instead. If you want to deploy a new Teams Connector in a different Azure region, see [Installing Teams Connectors in multiple Azure regions](#).
- The Microsoft Graph login you use must be a user in the tenant, not a Windows Live ID account.
- Before running your scripts we recommend that you check the Azure status (<https://status.azure.com>) for your region. Any ongoing issues in your region with the Azure services used by the Teams Connector could cause the script to fail.

Installation script

The following script is referred to as the **installation script**:

- This script only needs to be run **once** (per Azure subscription).
- As there are several commands in this installation process, we recommend running each group of commands step-by-step within PowerShell (you can copy-paste the commands below) to ensure that no elements are missed, and any unexpected issues are

identified. If you use PowerShell ISE instead of the normal PowerShell CLI prompt you can run one line at a time (select section and press F8).

- After launching PowerShell you must change directory to the folder into which you extracted the files from the Teams Connector ZIP.
- The `create_vmss_deployment.ps1` command in the script that creates the Teams Connector VMs can take up to 30 minutes to complete.
- See [Troubleshooting Microsoft Teams and Pexip Infinity integrations](#) for help with any installation script errors.

 Remember to run the variable initialization script (above) first, before running this installation script.

```
# Ensure the correct script and software combination is being used
try {$PxConnMajorVersion = (Get-Content .\version.json -ErrorAction Stop | Out-String | ConvertFrom-Json).major} catch {Write-Warning "Can't find version.json file. Make sure you run the installation script from the folder into which you extracted the files from the Teams Connector ZIP"}

if ($PxConnMajorVersion -ne "33"){Write-Warning "The Connector version (extracted ZIP files) and this deployment script version do not match. Connector version = $PxConnMajorVersion. Deployment script version = 33"}

# Connect to Microsoft Graph, Azure Resource Manager account and import Pexip CVI module
# Microsoft Graph commands
# Connect to Microsoft Graph
# If AAD/365 admin account is not the same as Azure Resource Manager admin account,
# the next section is to be run by the AAD admin.
#
# IMPORTANT: The output of IDs/credentials here must be saved as it will be required later
#
# IMPORTANT - DEPRECATION NOTICE:
# A certificate-based authentication process for the CVI application will be default for the Teams Connector in v34
# The password authentication method will be deprecated in a future release

# Set execution policy for the current PowerShell process, when prompted type A (Yes to All)
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# The Unblock-File cmdlet lets you run PowerShell script files that were downloaded from the internet.
# By default, these files are blocked to protect the computer from untrusted files.
Get-ChildItem -Recurse | Unblock-File

# Connect to Azure with an authenticated account for use with Azure Resource Manager (in same window to reuse variables)
Connect-AzAccount

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph

# Create Pexip CVI Application
# Create App
$App = New-PexTeamsCviApplication -AppDisplayName "$($PxBaseConnName)TeamsConn" -Confirm:$false
$AppId = $App.AppId

# Create App Password
$AppPassword = ($App | New-PexTeamsCviApplicationPasswordCredential -DisplayName "$($PxBaseConnName)TeamsConn").SecretText

Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host "### CVI App ID and credentials MUST be saved in the redeploy script ###"
Write-Host
Write-Host "### IMPORTANT - DEPRECATION NOTICE:"
Write-Host "### A certificate-based authentication process for the CVI app will be default for the Teams Connector in v34"
Write-Host "### The password authentication method will be deprecated in a future release"
Write-Host
Write-Host "``$AppId = ``$($AppId)``"
Write-Host "``$AppPassword = ``$($AppPassword)``"
Write-Host
```

```

Write-Host "`n-----`n"
Write-Host
Write-Host

# Change context to the Pexip Subscription and set the app credentials
Set-AzContext -SubscriptionId $PxSubscriptionId

$AppSecurePassword = ConvertTo-SecureString -AsPlainText $AppPassword -Force

# Azure Bot for the CVI AppID
# Create bot (must be globally unique, and only needs to be created once - in any of your regions)
Register-PexTeamsCviApplicationBot -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxBotResourceGroupName -BotName "(${PxBaseConnName})-TeamsBot" -AppId $AppId -Confirm:$false -Tag $tags

# Deploy Pexip Teams Connector admin consent web page
$AdminConsentUrl = Publish-PexTeamsAdminConsentWebSite -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxBotResourceGroupName -PxBaseConnName $PxBaseConnName -AppId $AppId -SourceAddressPrefixes $PxConsentSourceAddressPrefixes -Confirm:$false -Tag $tags

# Virtual Machine Scale Set (VMSS) creation
# Provide credentials to be used as local user/password for Pexip Teams Connector VMs
# Create a password (using the initialization script variables) for the Windows VM
$PxWinAdminSecurePassword = ConvertTo-SecureString -AsPlainText $PxWinAdminPassword -Force
$PxWinAdminCred = New-Object System.Management.Automation.PSCredential -ArgumentList $PxWinAdminUser,$PxWinAdminSecurePassword

# Optionally if you did not want to specify the password as a variable, use Get-Credential
# $PxWinAdminCred = Get-Credential

# Deploy the Teams Connector VMs
# this step can take up to 30 minutes to complete
.\create_vmss_deployment.ps1 -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxTeamsConnResourceGroupName -VmssName "(${PxBaseConnName})(${PxVmssRegion})" -VMAdminCredential $PxWinAdminCred -PfxPath $PxPfxCertFileName -TeamsConnectorFqdn $PxTeamsConnFqdn -PexipFqdns $PxNodeFqdns -instanceCount $PxTeamsConnInstanceCount -AppId $AppId -AppPassword $AppSecurePassword -StaticResourcesResourceGroupName $PxTeamsConnStaticResourceGroupName -IncidentReporting $PxTeamsConnIncidentReporting -RdpSourceAddressPrefixes $PxMgmtSrcAddrPrefixes -PexipSourceAddressPrefixes $PxNodesSourceAddressPrefixes -WupdScheduledInstallDay $PxWupdScheduledInstallDay -WupdScheduledInstallTime $PxWupdScheduledInstallTime -WupdActiveHoursStart $PxWupdActiveHoursStart -WupdActiveHoursEnd $PxWupdActiveHoursEnd -CustomerUsageAttribution $PxCustomerUsageAttribution -UseAzureHybridBenefit $PxUseAzureHybridBenefit -Tag $tags -TeamsConnectorApiApplicationId $TeamsConnectorApiApplicationId -FunctionsDedicatedHostingPlan $FunctionsDedicatedHostingPlan -EventHubSourceAddressPrefixes $EventHubSourceAddressPrefixes -VnetIntegration $VnetIntegration -PexipConfiguredConnectorFqdn $PexipConfiguredConnectorFqdn -PexipOutboundFqdn $PexipOutboundFqdn -ExistingVNETResourceId $PxExistingVNETResourceId -UsePrivateRouting $PxUsePrivateRouting

# supply the PFX certificate file password when prompted

# Please enter the password for the PFX certificate '.\xxxxxxx.pfx': *****

# Generating the next steps summary (this assumes you are connected to Microsoft Graph and with an authenticated account for use with Azure Resource Manager)
#
# Setting subscription
Set-AzContext -SubscriptionId $PxSubscriptionId

# Getting IP configurations
if ($PxUsePrivateRouting) {
    $LB = Get-AzLoadBalancer -ResourceGroupName $PxTeamsConnResourceGroupName
    $ExtLB = $LB | Where-Object { $_.Name.EndsWith("-LB") }
    $IntLB = $LB | Where-Object { $_.Name.EndsWith("-INTLB") }
    $LBExtIPID = $ExtLB.FrontendIpConfigurations[0].PublicIpAddress.id
    $LBIntIP = $IntLB.FrontendIpConfigurations[0].PrivateIpAddress
    $PublicIPAddresses = Get-AzPublicIpAddress -ResourceGroupName $PxTeamsConnStaticResourceGroupName
    $ConnectorPublicIP = $PublicIPAddresses | Where-Object Id -eq $LBExtIPID
    $publicIpAddress = $ConnectorPublicIP[0].IpAddress
    $privateIpAddress = $LBIntIP
} else {
    $LB = (Get-AzLoadBalancer -ResourceGroupName $PxTeamsConnResourceGroupName)[0]
    $LBPublicIPID = $LB.FrontendIpConfigurations[0].PublicIpAddress.id
    $PublicIPAddresses = Get-AzPublicIpAddress -ResourceGroupName $PxTeamsConnStaticResourceGroupName
    $ConnectorPublicIP = $PublicIPAddresses | Where-Object Id -eq $LBPublicIPID
    $publicIpAddress = $ConnectorPublicIP[0].IpAddress
}

# Getting connection string for the newly deployed Event hub

```

```

$eventHub = (Get-AzResource -ResourceGroupName $PxTeamsConnStaticResourceGroupName -ResourceType Microsoft.EventHub/namespaces)[0]
$eventHubKey = Get-AzEventHubKey -Name "pexip_teams_connector_access" -NamespaceName $eventHub.Name -ResourceGroupName
$eventHub.ResourceGroupName

# Printing next steps
Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host "When the Teams Connector is deployed, you have to create a DNS A record for your hostname,"
Write-Host "then the Office 365 admin must consent for the CVI App Id to join Teams Meetings"
Write-Host
Write-Host "1) Set up a public DNS A record for $($PxTeamsConnFqdn) pointing to the Public IP of "
Write-Host "    the load balancer $($publicIpAddress)"
Write-Host
Write-Host "2) Give consent to the CVI app. Go to: $AdminConsentUrl"
Write-Host
Write-Host "    If Management Consent Source Address prefixes are defined, the administrator"
Write-Host "    doing consent must come from one of these addresses (or subnets)."
```

```

Write-Host "    Consent address prefixes: $($PxConsentSourceAddressPrefixes)"
Write-Host
Write-Host "3) Update the Management Node setting 'Azure Event Hub connection string' for $($PxTeamsConnFqdn) to:"
Write-Host "    $($eventHubKey.PrimaryConnectionString)"
Write-Host
if ($PxUsePrivateRouting) {
    Write-Host "4) Set up a private DNS A record for $($PxTeamsConnFqdn) pointing to the (private) frontend IP of "
```

```

    Write-Host "    the internal load balancer $($privateIpAddress)"
    Write-Host "    See: https://docs.pexip.com/admin/teams\_routing.htm#enabling"
}
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host

```

```

# This script only applies to GCC High / Azure US Government Cloud deployments

# Ensure the correct script and software combination is being used
try {$PxConnMajorVersion = (Get-Content .\version.json -ErrorAction Stop | Out-String | ConvertFrom-Json).major} catch {Write-Warning "Can't
find version.json file. Make sure you run the installation script from the folder into which you extracted the files from the Teams Connector
ZIP"}

if ($PxConnMajorVersion -ne "33"){Write-Warning "The Connector version (extracted ZIP files) and this deployment script version do not match.
Connector version = $PxConnMajorVersion. Deployment script version = 33"}

# Set VmImage variable to hold the CIS STIG image properties - STIG image is optional but typical
# In a later step in this script you can choose not to use the STIG image
$VmImage = @{
    "sku"      = "cis-win-2019-stig"
    "offer"    = "cis-win-2019-stig"
    "publisher" = "center-for-internet-security-inc"
    "version"  = "latest"}

# Connect to Microsoft Graph, Azure Resource Manager account and import Pexip CVI module
# Microsoft Graph commands
# Connect to Microsoft Graph
# If AAD/365 admin account is not the same as Azure Resource Manager admin account,
# the next section is to be run by the AAD admin.
#
# IMPORTANT: The output of IDs/credentials here must be saved as it will be required later
#
# IMPORTANT - DEPRECATION NOTICE:
# A certificate-based authentication process for the CVI application will be default for the Teams Connector in v34
# The password authentication method will be deprecated in a future release

# Set execution policy for the current PowerShell process, when prompted type A (Yes to All)
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# The Unblock-File cmdlet lets you run PowerShell script files that were downloaded from the internet.
```



```
# By default, these files are blocked to protect the computer from untrusted files.
Get-ChildItem -Recurse | Unblock-File

# Connect to Azure USGovernment with an authenticated account for use with Azure Resource Manager (in same window to reuse variables)
Connect-AzAccount -EnvironmentName AzureUSGovernment

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph

# Create Pexip CVI Application
# Create App
$App = New-PexTeamsCviApplication -AppDisplayName "$($PxBaseConnName)TeamsConn" -Confirm:$false
$AppId = $App.AppId

# Create App Password
$AppPassword = ($App | New-PexTeamsCviApplicationPasswordCredential -DisplayName "$($PxBaseConnName)TeamsConn").SecretText

Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host "### CVI App ID and credentials MUST be saved in the redeploy script ###"
Write-Host
Write-Host "### IMPORTANT - DEPRECATION NOTICE:"
Write-Host "### A certificate-based authentication process for the CVI app will be default for the Teams Connector in v34"
Write-Host "### The password authentication method will be deprecated in a future release"
Write-Host
Write-Host "`$AppId = `"$($AppId)`""
Write-Host "`$AppPassword = `"$($AppPassword)`""
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host

# Change context to the Pexip Subscription and set the app credentials
Set-AzContext -SubscriptionId $PxSubscriptionId

$AppSecurePassword = ConvertTo-SecureString $AppPassword -Force

# Azure Bot for the CVI AppID
# Create bot (must be globally unique, and only needs to be created once - in any of your regions)
Register-PexTeamsCviApplicationBot -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxBotResourceGroupName -BotName "$($PxBaseConnName)-TeamsBot" -AppId $AppId -Confirm:$false -Tag $tags -TeamsEnvironmentName TeamsGCCHigh

# Deploy Pexip Teams Connector admin consent web page
$AdminConsentUrl = Publish-PexTeamsAdminConsentWebSite -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxBotResourceGroupName -PxBaseConnName $PxBaseConnName -AppId $AppId -SourceAddressPrefixes $PxConsentSourceAddressPrefixes -Confirm:$false -Tag $tags

# Virtual Machine Scale Set (VMSS) creation
# Provide credentials to be used as local user/password for Pexip Teams Connector VMs
# Create a password (using the initialization script variables) for the Windows VM
$PxWinAdminSecurePassword = ConvertTo-SecureString $PxWinAdminPassword -Force
$PxWinAdminCred = New-Object System.Management.Automation.PSCredential -ArgumentList $PxWinAdminUser,$PxWinAdminSecurePassword

# Optionally if you did not want to specify the password as a variable, use Get-Credential
# $PxWinAdminCred = Get-Credential

# Deploy the Teams Connector VMs
# this step can take up to 30 minutes to complete
# if you are not using a STIG image then remove the following parameter from this command: -VmImage $VmImage
```

```
.\create_vmss_deployment.ps1 -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxTeamsConnResourceGroupName -VmssName
"$($PxBaseConnName)$($PxVmssRegion)" -VMAdminCredential $PxWinAdminCred -PfxPath $PxPfxCertFileName -TeamsConnectorFqdn $PxTeamsConnFqdn -
PexipFqdns $PxNodeFqdns -instanceCount $PxTeamsConnInstanceCount -AppId $AppId -AppPassword $AppSecurePassword -
StaticResourcesResourceGroupName $PxTeamsConnStaticResourceGroupName -IncidentReporting $PxTeamsConnIncidentReporting -RdpSourceAddressPrefixes
$PxMgmtSrcAddrPrefixes -PexipSourceAddressPrefixes $PxNodesSourceAddressPrefixes -WupdScheduledInstallDay $PxWupdScheduledInstallDay -
WupdScheduledInstallTime $PxWupdScheduledInstallTime -WupdActiveHoursStart $PxWupdActiveHoursStart -WupdActiveHoursEnd $PxWupdActiveHoursEnd -
CustomerUsageAttribution $PxCustomerUsageAttribution -UseAzureHybridBenefit $PxUseAzureHybridBenefit -Tag $tags -TeamsConnectorApiApplicationId
$TeamsConnectorApiApplicationId -FunctionsDedicatedHostingPlan $FunctionsDedicatedHostingPlan -EventHubSourceAddressPrefixes
$EventHubSourceAddressPrefixes -VnetIntegration $VnetIntegration -VmImage $VmImage -TeamsEnvironmentName TeamsGCCHigh -
PexipConfiguredConnectorFqdn $PexipConfiguredConnectorFqdn -PexipOutboundFqdn $PexipOutboundFqdn -ExistingVNETResourceId
$PxExistingVNETResourceId -UsePrivateRouting $PxUsePrivateRouting

# supply the PFX certificate file password when prompted

# Please enter the password for the PFX certificate '.\xxxxxxxx.pfx': *****

# Generating the next steps summary (this assumes you are connected to Microsoft Graph and with an authenticated account for use with Azure
Resource Manager)
#
# Setting subscription
Set-AzContext -SubscriptionId $PxSubscriptionId

# Getting IP configurations
if ($PxUsePrivateRouting) {
    $LB = Get-AzLoadBalancer -ResourceGroupName $PxTeamsConnResourceGroupName
    $ExtLB = $LB | Where-Object { $_.Name.EndsWith("-LB") }
    $IntLB = $LB | Where-Object { $_.Name.EndsWith("-INTLB") }
    $LBExtIPID = $ExtLB.FrontendIpConfigurations[0].PublicIpAddress.id
    $LBIntIP = $IntLB.FrontendIpConfigurations[0].PrivateIpAddress
    $PublicIPAddresses = Get-AzPublicIpAddress -ResourceGroupName $PxTeamsConnStaticResourceGroupName
    $ConnectorPublicIP = $PublicIPAddresses | Where-Object Id -eq $LBExtIPID
    $publicIpAddress = $ConnectorPublicIP[0].IpAddress
    $privateIpAddress = $LBIntIP
} else {
    $LB = (Get-AzLoadBalancer -ResourceGroupName $PxTeamsConnResourceGroupName)[0]
    $LBPublicIPID = $LB.FrontendIpConfigurations[0].PublicIpAddress.id
    $PublicIPAddresses = Get-AzPublicIpAddress -ResourceGroupName $PxTeamsConnStaticResourceGroupName
    $ConnectorPublicIP = $PublicIPAddresses | Where-Object Id -eq $LBPublicIPID
    $publicIpAddress = $ConnectorPublicIP[0].IpAddress
}

# Getting connection string for the newly deployed Event hub
$eventHub = (Get-AzResource -ResourceGroupName $PxTeamsConnStaticResourceGroupName -ResourceType Microsoft.EventHub/namespaces)[0]
$eventHubKey = Get-AzEventHubKey -Name "pexip_teams_connector_access" -NamespaceName $eventHub.Name -ResourceGroupName
$eventHub.ResourceGroupName

# Printing next steps
Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host "When the Teams Connector is deployed, you have to create a DNS A record for your hostname,"
Write-Host "then the Office 365 admin must consent for the CVI App Id to join Teams Meetings"
Write-Host
Write-Host "1) Set up a public DNS A record for $($PxTeamsConnFqdn) pointing to the Public IP of "
Write-Host "    the load balancer $($publicIpAddress)"
Write-Host
Write-Host "2) Give consent to the CVI app. Go to: $AdminConsentUrl"
Write-Host
Write-Host "    If Management Consent Source Address prefixes are defined, the administrator"
Write-Host "    doing consent must come from one of these addresses (or subnets)."
Write-Host "    Consent address prefixes: $($PxConsentSourceAddressPrefixes)"
Write-Host
Write-Host "3) Update the Management Node setting 'Azure Event Hub connection string' for $($PxTeamsConnFqdn) to:"
Write-Host "    $($eventHubKey.PrimaryConnectionString)"
Write-Host
if ($PxUsePrivateRouting) {
    Write-Host "4) Set up a private DNS A record for $($PxTeamsConnFqdn) pointing to the (private) frontend IP of "
    Write-Host "    the internal load balancer $($privateIpAddress)"
    Write-Host "    See: https://docs.pexip.com/admin/teams\_routing.htm#enabling"
}
```

```
}
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host
```

After deploying the (first) Teams Connector

1. When the script ran, it generated some output that listed the CVI App ID and credentials, similar to this:

```
### CVI App ID and credentials MUST be saved in the redeploy script ###

### IMPORTANT - DEPRECATION NOTICE:
### A certificate-based authentication process for the CVI app will be default for the Teams Connector in v34
### The password authentication method will be deprecated


$AppId = "c054d1cb-7961-48e1-b004-389xxx32"
$AppPassword = "vAuo2ZlwhBD00538dhuhfGFRH58gdreHhDhff7635fdHhTE64gHTDYwd66rEW77uRE383=="
```

2. Copy the two output lines that define the CVI App ID and password and paste them into a copy of the [redploy script](#), replacing the existing lines that say:


```
$AppId = ""
$AppPassword = ""
```

This means that if you need to run the redeploy script, you will not rerun the commands that imported the PowerShell module and created the app. Instead you will set the variables to the ID and password of the CVI app that you created the first time.

3. Make sure you save your edited version of the redeploy script in a safe place, as it will be needed when upgrading, or if you need to redeploy, or deploy in another region.

-  It is critical that you update and store the redeploy script with the CVI app ID and password to ensure that upgrades/redeploys can be done using the same CVI app ID. (Note that the App ID for the API app is a separate ID and is stored in the variable initialization script.)

From version 33 you can use certificate-based authentication (CBA) to authenticate the Teams Connector CVI application towards MS Graph. In version 33, CBA is optional and the previous password-based authentication method is still the default mechanism.

-  The CBA method will be the default and recommended mechanism in version 34. Password-based authentication will still be supported in version 34 but we plan to deprecate it in a future release, thus we recommend migrating to CBA as soon as practicable.

Update DNS with Teams Connector name and IP address

You must now update DNS with the IP address of your Teams Connector load balancer that fronts the VM scale set.

The end of the installation script produced instructions for the required DNS changes. If you have closed down PowerShell, you can rerun the script that sets the installation variables, connect to Azure Resource Manager and then rerun the last section of the deployment script that prints out the details for you, for example:

```
1) Set up a public DNS A record for pexip-teamsconn-eu.teams.example.com pointing to the Public IP of
the load balancer 40.115.xx.191
```

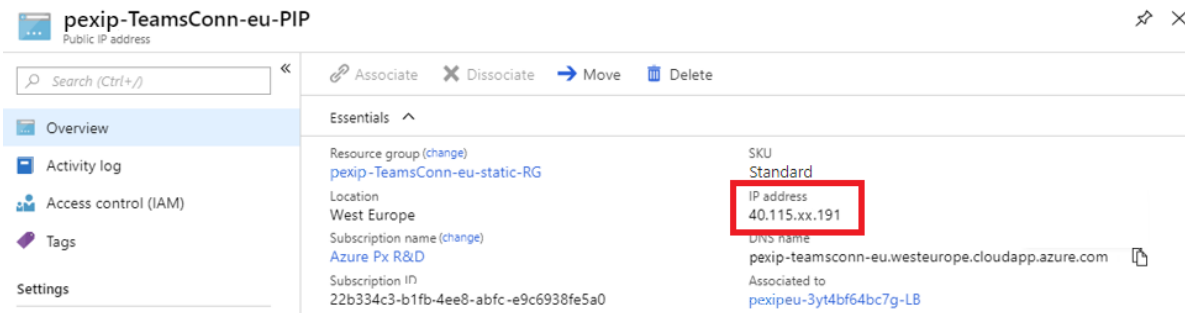
Within DNS you need to set up an A-record that links the Teams Connector hostname/FQDN to the IP address that was assigned by Azure to the load balancer. The A-record must be externally-resolvable.

The hostname of the Teams Connector is the name you used in the \$PxTeamsConnFqdn variable, for example "pexip-teamsconn-eu.teams.example.com".

The Azure-assigned IP address can also be obtained from the Azure portal:

1. Go to your subscription, then locate and select the resource group named <prefix>-TeamsConn-<region>-static-RG.
2. Select the item with a Type of Public IP address.

- The IP address is shown in the right-hand column.



The resulting DNS A-record you need to create, when using the example names from above, is:

NAME	TYPE	VALUE
pexip-teamsconn-eu.teams.example.com.	A	40.115.xx.191

- i** If you have enabled private routing you must also create an additional DNS record — see [Using private routing with the Teams Connector](#).

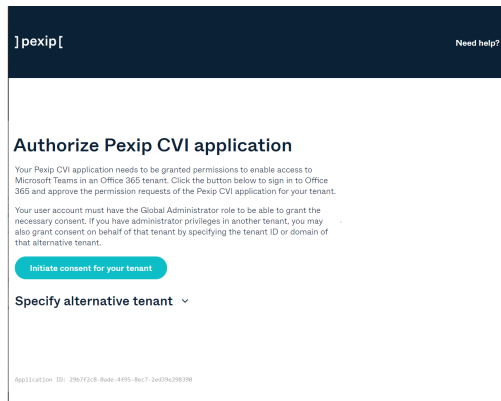
Authorize Pexip CVI application to join Teams meetings

Your Pexip CVI app needs to be granted permissions to enable access to Microsoft Teams meetings in an Office 365 tenant.

The end of the installation script produced instructions for the URL to visit to provide the necessary consent, for example:

2) Give consent to the CVI app. Go to: `https://yourname-pexip-cvi-abcde.azurewebsites.net/`

To provide consent to join Teams meetings:



- Use a web browser to go to the URL indicated in the output from the installation script and then follow the prompts in the consent wizard to authorize the CVI app.
- If the tenant you are logged in with is the tenant you will provide consent for, just select **Initiate the consent for your tenant**. If you have admin access for another tenant, you can specify an alternative tenant instead.
- You are asked to confirm the account. This must be an account with the Global Administrator role, to be able to grant the necessary consent.
- The [permissions](#) of the Pexip Teams Connector are listed. The domain shown here is the domain where the CVI App registration was created (this is the Azure Bot from the installation script) – it does not have to be the same Azure AD domain as where the Teams users are homed, but for most enterprises it will be the same domain.
- When admin consent is successfully granted, the success page is displayed.

We recommend saving the information shown on this page in case of future faultfinding to ensure full knowledge of which apps were consented in which tenant.

More information on CVI app permissions

Information about the permissions required by the Pexip CVI app is provided in the following table:

Permission	Details	More information
Read all users' full profiles	<p>The User.Read.All application permission provides access to properties and permissions for operations listed at https://docs.microsoft.com/en-us/graph/api/resources/users?view=graph-rest-1.0, in particular the methods and properties listed at https://docs.microsoft.com/en-us/graph/api/resources/user?view=graph-rest-1.0 that require the User.Read.All permission.</p> <p>It is used to:</p> <ul style="list-style-type: none"> Show the user's UPN (user@domain) instead of GUID in logs for an administrator to know which user was in which call. Display the photo of a user if the user has not started their video. <p>and will enable Pexip in future releases to:</p> <ul style="list-style-type: none"> Allow point-to-point call capabilities. 	<p>Application permissions: https://docs.microsoft.com/en-us/graph/permissions-reference#application-permissions-38</p>
Initiate outgoing 1 to 1 calls from the app	The Calls.Initiate.All application permission is used to support point-to-point calling scenarios.	<p>Calls permissions: https://docs.microsoft.com/en-us/graph/permissions-reference#calls-permissions</p>
Initiate outgoing group calls from the app	The Calls.InitiateGroupCall.All application permission is not currently used. It may be used in the future for creating ad hoc calls (e.g. to implement point-to-point calling scenarios).	
Join group calls and meetings as an app	The Calls.JoinGroupCall.All application permission allows VTC devices to join a Teams meeting as a trusted participant (permits lobby bypass).	
Join group calls and meetings as a guest	The Calls.JoinGroupCallAsGuest.All application permission allows VTC devices to join a Teams meeting as a guest/untrusted participant.	
Access media streams in a call as an app	The Calls.AccessMedia.All application permission is used to access the media streams of participants that are attending a Teams meeting.	
Read online meeting details	The OnlineMeetings.Read.All application permission is used to resolve the VTC meeting coordinate from the invite body into a Teams meeting.	<p>Online meetings permissions: https://docs.microsoft.com/en-us/graph/permissions-reference#online-meetings-permissions</p>
Sign in and read user profile	The User.Read delegated permission is added by default for Azure AD apps, but is not used by the Pexip Teams Connector.	<p>User permissions: https://docs.microsoft.com/en-us/graph/permissions-reference#user-permissions</p>

Any permissions that are not currently used are included in the current consent process as it is not practical to re-consent existing apps as new functionality becomes available via the SDK and/or as we add additional functionality to the Teams Connector itself.

Authorize CVI app to bypass Teams lobby and configure dialing instructions

The Pexip CVI app can be used to route callers directly into the Teams meeting (typically calls from trusted participants, such as internal employees or calls placed from registered and authenticated devices), or it can direct certain callers (such as external guests or calls placed from unknown devices) into the Teams lobby where they have to be admitted into the meeting by an existing participant in the meeting.

- Whether the caller is held in the lobby or routed directly into the meeting depends upon the **Treat as trusted** option that you can set when you configure your Call Routing Rules in Pexip Infinity. For example, if the rule only applies to calls received from registered endpoints then it will typically enable the **Treat as trusted** option which means that Pexip will route the call directly into the Teams meeting. If the **Treat as trusted** option is not enabled on the rule, Pexip routes the call into the meeting lobby. See [Call Routing Rules for direct and indirect routing](#) for more information.

Microsoft Teams meeting

Join on your computer or mobile app
[Click here to join the meeting](#)

Join with a video conferencing device
teams@example.com
Video Conference ID: 124 850 400 1
[Alternate VTC dialing instructions](#)

Note that if the Teams meeting option **Anonymous users can join a meeting** is turned off, all untrusted/guest VTCs are prevented from joining Teams calls, regardless of Pexip Infinity's rule settings.

- When a participant receives an invite to a Teams meeting, an "Alternate VTC dialing instructions" link to a webpage of alternative dialing addresses can be included. This provides customizable information for which address to dial, based on the type of client being used, such as a SIP device, an H.323 system or a browser, or whether you want to route callers via a Pexip IVR (Virtual Reception) into which they must enter the conference ID.

The PowerShell commands to configure the lobby bypass and dialing instructions require the Teams PowerShell module:

- Start a PowerShell session as Administrator and run the following commands:

```
Uninstall-Module MicrosoftTeams -AllVersions -Force
Install-Module MicrosoftTeams -MinimumVersion "4.0.0" -AllowClobber
```

(These commands ensure the appropriate MicrosoftTeams module is installed, and that it replaces any existing SkypeOnlineConnector module that may already be installed.)

- Run the following commands to sign in to your Teams tenant:

- In all standard deployments run:

```
Import-Module MicrosoftTeams
Connect-MicrosoftTeams
```

- If this is a GCC High / Azure US Government Cloud deployment then use these commands instead:

```
Import-Module MicrosoftTeams
Connect-MicrosoftTeams -TeamsEnvironmentName TeamsGCCH
```

Defining the CVI app behavior and grant interoperability

The `New-CsVideoInteropServiceProvider` PowerShell command is used to:

- define Pexip as your Cloud Video Interop service provider for Microsoft Teams
- allow the app to bypass the Teams lobby
- specify the content of the alternative dialing instructions.

To run the command, you need the **Global administrator** role.

The command takes the form:

```
New-CsVideoInteropServiceProvider -Name Pexip -TenantKey "<address>" -InstructionUri "<link>" -AllowAppGuestJoinsAsAuthenticated $true -AadApplicationIds "<App ID>"
```

which contains the following parameters:

- **Name:** this is a mandatory parameter and must be set to **Pexip**.
- **TenantKey:** this is the alias (SIP URI address) that you assign to the Pexip Virtual Reception that is to act as the IVR gateway into the Teams meetings. This must take the format `name@yourdomain`, for example `teams@example.com`.
See [Routing indirectly via a Virtual Reception \(IVR gateway\)](#) for more information on configuring the Pexip Virtual Reception.
- **AadApplicationIds:** when included, this is used in conjunction with setting `-AllowAppGuestJoinsAsAuthenticated $true` to allow the app to bypass the Teams lobby. Use the **CVI App ID** that was output by the installation script and copied into the redeploy script.
- **InstructionUri:** this is a link to a webpage of "Alternate VTC dialing instructions" that the recipient of the meeting invite can look at. The URI must include a set of parameters that control which information is displayed on the page. The URI takes the format:

```
https://<node_address>/teams/?conf={ConfId}&ivr=<alias>&d=<domain>&ip=<node_ip_address>&test=<test_call_alias>&prefix=<routing_rule_prefix>&w&qrcode
```

 where `<node_address>` is an FQDN that resolves to your Pexip Conferencing Nodes (it can also be the FQDN or IP address of an individual Conferencing Node). Note that:
 - To view this webpage, the client application used to view the invitation must be able to access the specified Conferencing Nodes (or alternative server) on HTTPS 443/TCP.
 - This FQDN is not necessarily the same name as that specified in the `$PxNodeFqdns` variable (as that is used only for external DNS resolution from the Teams Connector to your externally-facing nodes). You must ensure that the FQDN used here is resolvable by any internal or external client applications that may be used to view the invitation i.e. that they can access the webpage on the nodes referenced by the FQDN. This means that if these nodes have private addresses, then depending on your internal network routing, you may need appropriate local DNS resolution for the `<node_address>` FQDN for internally-based clients, in addition to external DNS resolution for that FQDN for externally-based clients.

The **InstructionUri** parameters are:


Parameter	Mandatory	Description
conf	Yes	This must be set to <code>{ConfId}</code> and when displayed it will contain the conference ID of the Teams meeting.
ivr	Yes	The name part of the alias of the Pexip Virtual Reception that you will configure on Pexip Infinity later, and that will act as the IVR gateway. Do not include the domain — this is the <code>d</code> parameter below. Note that <code><ivr@d></code> , e.g. <code>teams@example.com</code> , must match the name of the alias that you will configure for the Virtual Reception.
d	Yes	The domain name of your Pexip Infinity platform e.g. <code>example.com</code> . This is used as the domain for all of the URI-style addresses that are displayed on the webpage.
prefix	No	A prefix to apply to the <code>conf</code> parameter when building the address to call for direct routing into the Teams meeting. Typically you only need to use a prefix if you have a more complicated dial plan. If used, the prefix will typically match the Call Routing Rules set up in Pexip Infinity to route calls into Teams meetings.
w	No	Displays the "From a browser" access details on the webpage. There is no value associated with this parameter. Note that these details are not displayed if the user is viewing the page on Microsoft Edge (as it is expected they would join the Teams meeting directly via Teams itself).
ip	No	This is the public-facing IP address of one of your Conferencing Nodes. When specified, alternative direct dialing options via that IP address (which may be required by some H.323 systems for example) are included. You can only specify a single address. If included, the IP address specified here must also be added as an alias to the Virtual Reception that you will configure on Pexip Infinity later.
test	No	Includes a "Test call" option on the webpage, where the value of this parameter is the name part of the alias to dial e.g. <code>test_call</code> . Do not include the domain — this is the <code>d</code> parameter above. This uses Pexip Infinity's inbuilt Test Call Service; you must ensure that <code><test@d></code> , e.g. <code>test_call@example.com</code> , matches the name of the alias configured in Pexip Infinity for the test call service.

Parameter	Mandatory	Description
qrcode	No	Includes a QR code on the webpage, which when scanned by a device with a supported Connect app installed (such as Pexip Connect for RealWear or one of the Connect mobile apps) will open the meeting directly in that app. There is no value associated with this parameter.

An example `InstructionUri` value could be: `https://px.vc.example.com/teams/?conf={ConfId}&ivr=teams&d=example.com&ip=198.51.100.40&test=test_call&w&qrcode`

and that would produce the following webpage (where the Teams conference ID is "234567890"):

]pexip[
Video meeting invitation




Join the meeting directly

From a VTC/SIP system, enter: [234567890@example.com](tel:234567890@example.com)
Alternative addresses (e.g. H.323 systems): [234567890@198.51.100.40](tel:234567890@198.51.100.40) or [198.51.100.40#234567890](tel:198.51.100.40#234567890)


Join via lobby service

From a VTC/SIP system, enter: [teams@example.com](tel:teams@example.com) or [198.51.100.40](tel:198.51.100.40) then enter the conference ID: [234567890](tel:234567890) followed by #




From a browser

Go to: <https://px.vc.example.com/webapp/?conference=234567890@example.com>



From the Pexip app on a supported device:

Scan this with your camera: <pexip://234567890@example.com?host=px.vc.example.com>



Test call

To test your connection from a VTC/SIP system, enter: [test_call@example.com](tel:test_call@example.com) and verify that you can see and hear yourself

Powered by Pexip video interoperability for Microsoft Teams

Here is an example of the complete `New-CsVideoInteropServiceProvider` command:

```
New-CsVideoInteropServiceProvider -Name Pexip -TenantKey "teams@example.com" -InstructionUri
"https://px.vc.example.com/teams/?conf={ConfId}&ivr=teams&d=example.com&ip=198.51.100.40&test=test_call&w&qrcode" -
AllowAppGuestJoinsAsAuthenticated $true -AadApplicationIds "c054d1cb-7961-48e1-b004-389e81356232"
```

Use the following steps to assign lobby-bypass capabilities to the app, define the content of the "Alternate VTC dialing instructions" page, and grant interoperability for the users in your tenant.

i The following commands may take several hours (sometimes over 24 hours) to come into effect.

1. Assign lobby-bypass capabilities to the app and specify the "Alternate VTC dialing instructions". This command takes the form:

```
New-CsVideoInteropServiceProvider -Name Pexip -TenantKey "<address>" -InstructionUri "<link>" -
AllowAppGuestJoinsAsAuthenticated $true -AadApplicationIds "<App ID>"
```

For example (note that all of the available command parameters are described above):

```
New-CsVideoInteropServiceProvider -Name Pexip -TenantKey "teams@example.com" -InstructionUri
"https://px.vc.example.com/teams/?conf={ConfId}&ivr=teams&d=example.com&test=test_call&w&qrcode" -
AllowAppGuestJoinsAsAuthenticated $true -AadApplicationIds "c054d1cb-7961-48e1-b004-389e81356232"
```

2. Grant interoperability for the users in your tenant. To grant interoperability for all users:

```
Grant-CsTeamsVideoInteropServicePolicy -PolicyName PexipServiceProviderEnabled -Global
```

For testing purposes you can enable interop to named users by using the `-Identity` switch instead of `-Global`, for example:

```
Grant-CsTeamsVideoInteropServicePolicy -PolicyName PexipServiceProviderEnabled -Identity alice@example.com
```

Notes:

- The [troubleshooting](#) topic contains some PowerShell commands that are useful when troubleshooting or maintaining your system.
- To change the settings for your Teams meetings, such as customizing your meeting invitations, see <https://docs.microsoft.com/en-us/microsoftteams/meeting-settings-in-teams>.
- CVI participants can join meetings created using the "Meet Now" feature from within the Calendar page in the Teams client. However, meetings created using "Meet Now" from within the Teams/Channel page do not contain CVI join details.

Next steps

You must now complete the configuration within Pexip Infinity as described in [Configuring Pexip Infinity as a Microsoft Teams gateway](#).

Installing the Teams Connector using a blue-green deployment/upgrade strategy

The Teams Connector deployment/upgrade process supports a blue-green deployment strategy. This approach allows you to create separate environments where one environment (blue) is running the current application version and another environment (green) is running the new application version. This means that you can test both of these deployments separately, and switch between them.

This is different from the regular deployment and upgrade strategy that involves deploying a single Teams Connector environment that is destructively replaced during the upgrade process.

This topic covers:

- [Benefits of using a blue-green strategy](#)
- [How it works](#)
- [Deployment requirements and guidelines](#)
- [Installing the Teams Connectors](#)
- [Pexip Infinity Management Node configuration](#)
- [Upgrading the Teams Connector](#)

Benefits of using a blue-green strategy

We recommend using a blue-green deployment strategy, where you have two Teams Connector environments deployed in parallel, as it:

- Provides a non-destructive upgrade path.
- Increases application availability during the upgrade process.
- Enables upgrade activities to be done in business hours when access to required service administrators/owners are more readily available, before a planned "switch over" window.
- Reduces time-pressure and risk if there are delays due to Azure resources not being available.
- Reduces deployment risk by simplifying the rollback process if a deployment fails.

Advantages over alternative strategies

An alternative upgrade strategy that also enables you to maintain a working production environment while upgrading to a new software version would be to deploy a brand new Teams Connector for every software release and then switch to using that system when it is tested and ready.

This would work but the drawbacks to this approach are that for every upgrade you would need to involve all of the service administrators/owners and have to go through all of the other installation steps every time, such as creating DNS records, possibly issuing a new certificate, obtaining Azure permissions, creating a new API app etc. Whereas with a blue-green strategy you plan ahead and set up two systems at the time of the initial deployment, and keep and re-use the same resources — you just switch back and forth between the two environments.

How it works

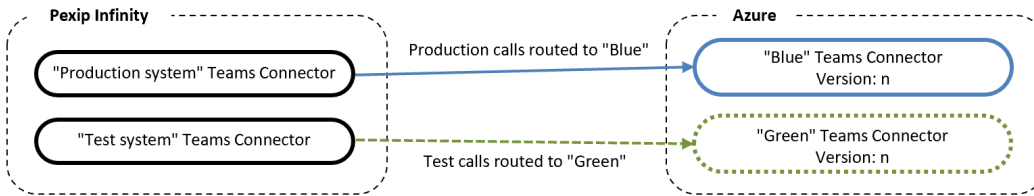
The main difference between the blue-green deployment and upgrade strategy described here compared to the regular deployment is that you deploy two Teams Connectors (one called "blue" and another called "green") and switch between them as you upgrade from one version of Teams Connector software to the next. Whereas with the regular deployment you only maintain one Teams Connector which you replace at every upgrade cycle.

Specifically, the differences between the two methods are:

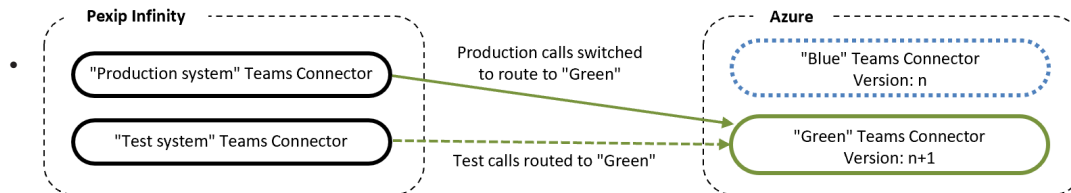
- For blue-green you deploy two Teams Connectors. Each Teams Connector needs a unique DNS name and a TLS certificate that matches that identity. You use different names for the `$PxVmssRegion` variable to create specific, separate Azure resources for the two deployments (you use a separate variables script for each deployment).

- For blue-green each deployment requires its own **API app**. However, the two deployments can both use the same **CVI app** and you only need to perform one app authorization process (per tenant). You also only need one Azure bot.
- With the regular deployment procedure:
 - You simply replace your existing single deployment every time you upgrade to the next Teams Connector software version. It reuses the existing Azure resources, DNS records and so on.
 - While upgrading you have a period of time when you have no Teams interop service and no ability to separately test the latest version. Typically you have to perform the entire upgrade process during "out-of-hours".

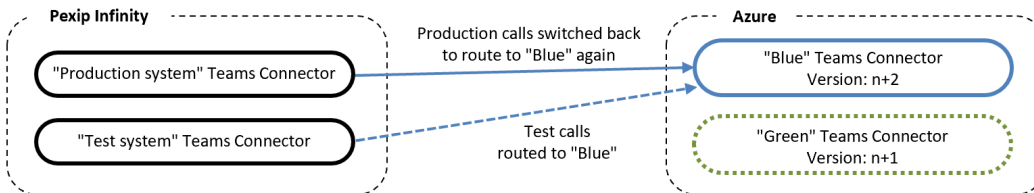
Initial Installation: Version n



After first upgrade: Version n+1



After second upgrade: Version n+2



With the blue-green procedure:

- You have one deployment which is currently active (in "production") and one that is dormant. When you upgrade, you upgrade the dormant system, without any interference to the active system. You can then test the new system and switchover to that new system when you are happy it is functioning as expected. The previously active system then becomes dormant until the next upgrade occasion.
- You have 2 DNS records — one for the "blue" system and one for the "green" system.
- Within Pexip Infinity we suggest that you configure 2 Teams Connector systems, using one for the active/production system (and this is used by all of your Call Routing Rules) and one for the dormant/test system (that is used by a "test" Call Routing Rule).
- At each switchover you toggle the addresses of the two Teams Connector systems that are configured in the Pexip Infinity Management Node i.e. they switch between pointing at the "blue" and "green" systems.

Deployment requirements and guidelines

The deployment steps are similar to a [regular](#) Teams Connector deployment. However, you should note that:

- You deploy two Teams Connectors. Each Teams Connector needs a unique DNS name and a TLS certificate that matches that identity. You can use a single certificate for both Teams Connectors; this can either be a wildcard certificate or a certificate that contains the Subject Alternative Name (altNames attribute) entries for both Teams Connectors.

- You **do not** need to create a new **CVI app** or perform any additional app authorizations when deploying the second Teams Connector — only one CVI app registration is required per tenant. However you **do** need to create a new **API app** for each new deployment.
- You **do not** need to create a new CVI app or API app when upgrading (redeploying) to a new version.
- Both Teams Connectors can use the same set of Conferencing Nodes (the `$PxNodeFqdns` variable in the initialization script).
- You create 2 DNS A-records (one for each deployment), and you need to enable the relevant firewall ports for both Teams Connector deployments while you are testing / switching over.

This guide assumes that this is your first Teams Connector deployment. If you have an existing Teams Connector and want to switch to using a blue-green upgrade strategy then you can still follow the principles explained in this guide by creating a second "green" deployment to use alongside your existing deployment (which in effect becomes your "blue" deployment, using whatever name is currently assigned to `$PxVmssRegion` in your existing variables script).

Installing the Teams Connectors

As with the [regular installation](#), you deploy Teams Connector through PowerShell ISE commands and scripts.

1. Perform these steps twice — one time for each deployment, where **<version>** is either "blue" or "green".
1. Create a new variable initialization script for your **<version>** deployment. You should do this by making a copy of the [variable initialization script](#). Save this script to a safe location as it will be needed for future upgrades.
2. Update the variables in the initialization scripts with the values for your **<version>**. The variables that must be different for the **<version>** deployment are:
 - `$PxVmssRegion`: the short name that identifies the deployment. In this case it refers to the Teams Connector deployment identifier: e.g. "blue" or "green". Note that you can combine this with a regional deployment if required, e.g. "eubblue".
 - `$PxTeamsConnFqdn`: the hostname of the Teams Connector, for example "pexip-teamsconn-blue-eu.teams.example.com" or "pexip-teamsconn-green-eu.teams.example.com".
 - `$TeamsConnectorApiApplicationId`: this must be updated to hold the ID of a new Teams Connector API app for the **<version>** deployment, and is described later in the process below.
 - `$PxExistingVNETResourceId`: if you are using [private routing](#), each deployment must use its own non-overlapping VNET.
 - If this is not your first deployment and you have already created the CVI application, you need to update `$AppId` and one of `$AppPassword` or `$AppCertificatePath` (see [Using certificate-based authentication for the Teams Connector CVI application](#)).

All other variables can be identical across both scripts.

3. Create the static and dynamic resource groups for the **<version>** deployment and ensure that the person performing the installation has the **Owner** role for them.
 - a. These steps **must** be performed by the **Owner** of the Azure subscription used for the Teams Connector.
 - a. Run the following PowerShell command to connect to Azure:
 - In all standard deployments run:
`Connect-AzAccount`
 - If this is a GCC High / Azure US Government Cloud deployment then use this command instead:
`Connect-AzAccount -EnvironmentName AzureUSGovernment`
 Then follow the prompts to sign in to Azure.
 - b. Run the variable initialization script for the **<version>** deployment (to set the required subscription, resource group name and region variables).
 - c. Ensure that you are using the Azure subscription for the Teams Connector:
`Set-AzContext -SubscriptionId $PxSubscriptionId`
 - d. Run the following script to create the resource groups:
 - ❗ You only need to run the command that creates the resource group for the Azure Bot (`$PxBotResourceGroupName`) once — it can be shared between all **<version>** deployments.

```
# Resource group for static resources for the Teams Connector / region
New-AzResourceGroup -Name $PxTeamsConnStaticResourceGroupName -Location $PxAzureLocation -Force -Tag $tags
```

```
# Resource group for the Teams Connector VMSS (per region)
New-AzResourceGroup -Name $PxTeamsConnResourceGroupName -Location $PxAzureLocation -Tag $tags

# Resource group for the Azure Bot
New-AzResourceGroup -Name $PxBotResourceGroupName -Location $PxAzureLocation -Tag $tags
```

- e. Ensure that the person who will perform all of the remaining installation steps has the Azure **Owner** role for the static and dynamic resource groups, and **Contributor** role for the Azure Bot resource group you have just created. If the Owner of the Azure subscription will perform all of the remaining installation steps then you can skip to [Create the Teams Connector API app](#) below.

Otherwise, you must run the following commands to assign the required roles for the **resource groups** you have just created to the person who will perform all of the remaining installation steps:

```
New-AzRoleAssignment -SignInName <email_name> -RoleDefinitionName "Owner" -ResourceGroupName
$PxTeamsConnStaticResourceGroupName
New-AzRoleAssignment -SignInName <email_name> -RoleDefinitionName "Owner" -ResourceGroupName $PxTeamsConnResourceGroupName
New-AzRoleAssignment -SignInName <email_name> -RoleDefinitionName "Contributor" -ResourceGroupName $PxBotResourceGroupName
```

where **<email_name>** is the email address / user principal name of the person who will perform the remaining steps; for example where `alice@example.com` will perform the upgrade/redeploy, the `SignInName` would be `-SignInName`

`alice@example.com` for the commands listed above.

Note:

- Some of these resource groups only need creating once; others need recreating when you upgrade or if you redeploy:
 - The static resource group (`$PxTeamsConnStaticResourceGroupName`) only has to be created when deploying for the first time — you do not have to repeat this when redeploying or for subsequent upgrades to either deployment.
 - The dynamic resource group (`$PxTeamsConnResourceGroupName`) has to be recreated whenever you upgrade or redeploy.
- In the future, if another person were to upgrade or redeploy the Teams Connector, that person would also have to be granted the appropriate roles for these resource groups.
- You can also use the Azure portal to check/assign permissions by selecting the resource group and using the **Access control (IAM)** option.

4. Create the Teams Connector API app.

You must create an Azure app that is used to secure requests to the Teams Connector APIs.

- i** Each deployment must have its own Teams Connector API app, and thus the variables initialization script for each deployment must be updated to contain the API App ID for that deployment.

- a. Run the following PowerShell command to connect to Azure:

- In all standard deployments run:


```
Connect-AzAccount
```
- If this is a GCC High / Azure US Government Cloud deployment then use this command instead:


```
Connect-AzAccount -EnvironmentName AzureUSGovernment
```

Then follow the prompts to sign in to Azure.

- b. Change directory to the folder into which you extracted the files from the Teams Connector ZIP.
- c. Run the following PowerShell commands to connect to Microsoft Graph:

```
# Set execution policy for the current PowerShell process, when prompted type A (Yes to All)
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# The Unblock-File cmdlet lets you run PowerShell script files that were downloaded from the internet.
# By default, these files are blocked to protect the computer from untrusted files.
Get-ChildItem -Recurse | Unblock-File

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph
```

- d. Run your variable initialization script for the **<version>** deployment to set the required variables.
- e. Run the following command to create the Teams Connector API app.

```
$teamsConnectorApiApp = New-MgApplication -DisplayName "${PxBaseConnName}-TeamsConn-${PxVmssRegion} Pexip Teams Connector API" -
SignInAudience "AzureADMyOrg"
```

Please allow one minute for this command to complete and propagate within Azure.

- f. Run the following commands to obtain the Teams Connector API app ID.

Note that if the **New-MgServicePrincipal** command fails, this means that you have not waited long enough for the previous command to complete.

```
$teamsConnectorApiSp = New-MgServicePrincipal -AppId $teamsConnectorApiApp.AppId -Tags {WindowsAzureActiveDirectoryIntegratedApp}
$TeamsConnectorApiApplicationId = $teamsConnectorApiApp.AppId

Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host "### Teams Connector API App ID MUST be saved in the variables initialization script ###"
Write-Host
Write-Host "`$TeamsConnectorApiApplicationId = `"$($TeamsConnectorApiApplicationId)`""
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host
```

- g. When the command runs, it generates some output that lists the Teams Connector API App ID, similar to this:

```
### Teams Connector API App ID MUST be saved in the variables initialization script ###
$TeamsConnectorApiApplicationId = "36ee4c6c-0812-40a2-b820-b22ebd02bce4"
```

- h. Copy the output line that defines the API App ID and paste it into the variable initialization script, replacing the existing line in the script that says:

```
$TeamsConnectorApiApplicationId = ""
```

(If you will be performing the rest of the deployment, in the same PowerShell session, there is no need to re-run the variable initialization script as the new \$TeamsConnectorApiApplicationId variable has been set in the previous steps.)

- i. If somebody else will be completing the deployment, ensure that the person who will perform all of the remaining installation steps has **Owner** permissions for the API app. See [Assigning Owner permissions for the API app](#) for more information.

Note that:

- This app does not have to be granted any permissions. It does not have access to any resources in the Azure AD tenant. It has no associated credentials.

5. Run the [standard installation script](#) to install the <version> Teams Connector deployment.

Note that the following steps in the installation script only have to be run **once** — do not perform them a second time i.e. when installing the second <version> or if you are re-using another pre-existing CVI application:

- Creating the CVI application and its associated credentials (but you must still run the command to assign the \$AppSecurePassword variable if you are using password-based authentication).
- Creating the Azure Bot.
- Deploying the Teams Connector admin consent web page.

6. Update DNS with the <version> Teams Connector FQDN (DNS name) and IP address. The deployment script should have printed the instructions. Thus, at the end of the process you will have two DNS A-records — one for each <version>.

Post deployment steps

After successful initial deployment and testing of both of these Teams Connectors, you can remove the dynamic resource group (\$PxTeamsConnResourceGroupName) of the **green** deployment so you don't have to pay for the redundant Azure resources while that environment is not in use.

(When you upgrade to the next version you will redeploy the **green** deployment and you will then have two functioning deployments again — the old version and the new version.)

Pexip Infinity Management Node configuration

This section describes how you configure the two Teams Connectors in the Pexip Infinity Management Node, covering the [installation](#), [upgrade](#) and [switchover](#) scenarios.

Initial installation

When installing the Teams Connectors you can follow the standard instructions to configure your Pexip Infinity platform ([Configuring Pexip Infinity as a Microsoft Teams gateway](#)).

However, as you are configuring two Teams Connectors within Pexip Infinity, we suggest that:

1. When defining your Teams Connectors (**Call Control > Microsoft Teams Connectors**):
 - Define one Teams Connector for the "blue" deployment that is named, for example, "Production system" and is configured with the address and Event Hub connection details for the first/blue deployment.
 - Define a second Teams Connector for the "green" deployment that is named, for example, "Test system" and is configured with the address and Event Hub connection details for the second/green deployment. You should also then deselect the **Enable Azure Event Hub** checkbox at this stage to prevent any unnecessary alarms being raised while the test system is not in use.
2. When defining your Call Routing Rules and system locations:
 - Configure all of your main rules and locations to use the "Production system" Teams Connector.
 - Configure one Call Routing Rule to use for testing purposes. For example, you could configure the rule as follows:
 - **Destination alias regex match:** `teamstesting\.{0,12}(@example\.com)?`
(replace example\.com with your own domain)
 - **Destination alias regex replace string:** `\1`
 - **Call target:** select the "Test system" Teams Connector

This means whenever you want to use your test system you can dial a prefix of "teamstesting." in front of the ID of the Teams conference you want to join, and the call will then be routed through the test Teams Connector.

Upgrade and test

After you have upgraded your test system to the latest Teams Connector software, you can test the new software without having to make any significant configuration changes within Pexip Infinity — the only configuration change required is to select the **Enable Azure Event Hub** checkbox on the "Test system" Teams Connector for the duration of the testing period.

- Assuming "blue" is the current production system, after upgrading the "green" system to the latest software you can test it simply by using the existing "test" rule (that you set up during the initial installation and is configured to use the "Test system" Teams Connector).
- Your existing production system will continue to work normally (using the other Call Routing Rules and the "Production" Teams Connector).

Pexip Infinity version compatibility

Normally you should ensure that you are running the same software version (including minor releases) of Pexip Infinity and Teams Connector to ensure compatibility between the two platforms.

When testing the new Teams Connector software while also running the existing production version it is likely that, for a period of time, you will have a version mismatch between the two platforms. Typically when upgrading to a new minor release there should not be any incompatibility issues. However when upgrading to a new major version of the Teams Connector there could be compatibility issues, and any new features introduced in that new version may not work as expected, and possibly the interop service may not work at all. You must always read the release notes for any known compatibility issues and upgrade guidelines. In some cases you may need to deploy and use a test Pexip Infinity platform that is also running the same software version as the new Teams Connector.

Switching over to the new Teams Connector software

After you have upgraded and tested the new Teams Connector software, you can switch the configuration of the two deployments (**Call Control > Microsoft Teams Connectors**).

Assuming "blue" is the current production system:

1. Update the "Production system" Teams Connector to use the address and Event Hub connection details of the "green" deployment.
You are now live with the new software and any new calls will be routed through the new "green" system (any existing calls will remain connected via the "blue" deployment).
2. Update the "Test system" Teams Connector with the address and Event Hub connection details of the "blue" deployment. You should then also deselect the **Enable Azure Event Hub** checkbox again.
This gets it ready for the next upgrade cycle.

At the next upgrade, just repeat the process of switching the configuration of the two Teams Connectors — this time the "blue" system will go back to being the production environment, and "green" will go back to being the test environment. And then just repeat this toggling process in the future.

The benefit of this approach is that you only have to update the configuration of the two Teams Connectors whenever you want to complete the switchover, rather than potentially having to update lots of Call Routing Rules to switch between the two Teams Connector deployments. However, any process that follows the same principles of toggling between the two deployed Teams Connectors is fine.

If you do need to fallback to the previous system you can just reverse the configuration steps to reinstate the previous blue/green deployment as the production system.

Note that no DNS changes required when switching over (after initially creating the 2 A-records during the installation process).

Upgrading the Teams Connector

When you want to upgrade to the latest Teams Connector version you should:

1. Redeploy the deployment which is currently not in use. For example, after initially using the **blue** system you will redeploy the **green** system. You will then have two functioning deployments again — the old version and the new version.
2. Test the newly redeployed system.
3. Switch over to the new system and decommission the old deployment version when the transition is complete.

Upgrades to a new version are performed by following the standard steps outlined in [Upgrading the Teams Connector to the latest software](#). However, when using the blue-green deployment strategy note that:

- You always upgrade the Teams Connector deployment that is currently not in use. The variable script you use for the redeployment has to match the variable script used to deploy that particular deployment (i.e. **blue** or **green**).
- As with any upgrade of the Teams Connector you need to ensure that the dynamic resource group for that deployment is empty. This means that when redeploying, for example the **blue** deployment, you need to delete and recreate the `$PxTeamsConnResourceGroupName` of the **blue** deployment. Make sure that the name doesn't change when recreating the resource group — it must stay the same as when used previously for that version.
- After the deployment finishes, and has been tested, follow the Management Node configuration described [above](#) to complete the switchover.
- When the transition is complete you can optionally decommission the old deployment version by removing the dynamic resource group (`$PxTeamsConnResourceGroupName`) of the old/previous deployment so you don't have to pay for the redundant Azure resources while that environment is not in use.

Installing Teams Connectors in multiple Azure regions

Large enterprises may want to install a Teams Connector in multiple regions to provide local capacity/resources.

As with your initial Teams Connector installation, you must follow the same guidelines when deploying additional Teams Connectors:

- The Azure region for the new Teams Connector must support Automation and Fs series instance types. Ensure that you have sufficient resource quota and capacity for those instance types in that region.
 - Each Teams Connector that you deploy needs a unique DNS name and a certificate that matches that identity.
 - You may also be deploying a new pool of Conferencing Nodes to use with the new Teams Connector. These nodes would typically be in the same Azure region as the new Teams Connector, or in an on-premises datacenter in the same geographic area. You may want to use a different certificate (with a different pool name) for this set of Conferencing Nodes.
- i** You do **not** need to create a new CVI app or perform any additional app authorizations when installing additional Teams Connectors — only one CVI app registration is required per tenant. However you do need to create a new API app for each new region, as described here.

Installing the additional Teams Connector application into Azure

As with the initial installation, you deploy additional Teams Connector applications through PowerShell ISE commands and scripts:

1. Create a new variables initialization script for the new Azure region. You should do this by making a copy of your existing initialization script.
2. Update the regional variables in the new initialization script with the values for your new Azure region. The variables you need to update are:
 - **\$PxVmssRegion**: the short name that represents the new region in which you are deploying the new Teams Connector.
 - **\$PxTeamsConnFqdn**: the hostname of the new Teams Connector.
 - **\$PxNodeFqdns**: the pool name or names of the Conferencing Nodes that will communicate with the new Teams Connector.
 - **\$PxAzureLocation**: the name of the Azure region into which you are deploying the new Teams Connector.
 - **\$PxPfxCertFileName**: the filename of the PFX certificate file to upload to the new Teams Connector.
 - **\$PxNodesSourceAddressPrefixes**: the IP addresses of the Conferencing Nodes that will communicate with the new Teams Connector instances.
 - **\$TeamsConnectorApiApplicationId**: this must be updated to hold the ID of a new Teams Connector API app for this region, and is described later in the process below.
3. Create the static and dynamic resource groups for the new region and ensure that the person performing the installation has the **Owner** role for them. (You do not need to create a resource group for the Azure Bot.)

i These steps **must** be performed by the **Owner** of the Azure subscription used for the Teams Connector.

 - a. Run the following PowerShell command to connect to Azure:
 - In all standard deployments run:


```
Connect-AzAccount
```
 - If this is a GCC High / Azure US Government Cloud deployment then use this command instead:


```
Connect-AzAccount -EnvironmentName AzureUSGovernment
```

Then follow the prompts to sign in to Azure.

- b. Run the variable initialization script (to set the required subscription, resource group name and region variables).
- c. Ensure that you are using the Azure subscription for the Teams Connector:


```
Set-AzContext -SubscriptionId $PxSubscriptionId
```
- d. Run the following script to create the resource groups for static and dynamic resources:

```
# Resource group for static resources for the Teams Connector / region
New-AzResourceGroup -Name $PxTeamsConnStaticResourceGroupName -Location $PxAzureLocation -Force -Tag $tags

# Resource group for the Teams Connector VMSS (per region)
New-AzResourceGroup -Name $PxTeamsConnResourceGroupName -Location $PxAzureLocation -Tag $tags
```

- e. Ensure that the person who will perform all of the remaining installation steps has the Azure **Owner** role for the static and dynamic resource groups you have just created. If the Owner of the Azure subscription will perform all of the remaining installation steps then you can skip to [Create the Teams Connector API app](#), below.

Otherwise, you must run the following commands to assign the required roles for the **resource groups** you have just created to the person who will perform all of the remaining installation steps:

```
New-AzRoleAssignment -SignInName <email_name> -RoleDefinitionName "Owner" -ResourceGroupName $PxTeamsConnStaticResourceGroupName
New-AzRoleAssignment -SignInName <email_name> -RoleDefinitionName "Owner" -ResourceGroupName $PxTeamsConnResourceGroupName
```

where **<email_name>** is the email address / user principal name of the person who will perform the remaining steps; for example where `alice@example.com` will perform the upgrade/redeploy, the `SignInName` would be `-SignInName alice@example.com` for the commands listed above.

Note:

- Some of these resource groups only need creating once; others need recreating when you upgrade or if you redeploy:
 - The static resource group (`$PxTeamsConnStaticResourceGroupName`) only has to be created when deploying in a region for the first time — you do not have to repeat this when redeploying or for subsequent upgrades.
 - The dynamic resource group (`$PxTeamsConnResourceGroupName`) has to be recreated in each region whenever you upgrade or redeploy.
- In the future, if another person were to upgrade or redeploy the Teams Connector, that person would also have to be granted the appropriate roles for these resource groups.
- You can also use the Azure portal to check/assign permissions by selecting the resource group and using the **Access control (IAM)** option.

4. Create the Teams Connector API app.

You must create in this region an Azure app that is used to secure requests to the Teams Connector APIs.

- ❗ Each region must have its own Teams Connector API app, and thus the variables initialization script for each region must be updated to contain the API App ID for that region.

- a. Run the following PowerShell command to connect to Azure:

- In all standard deployments run:


```
Connect-AzAccount
```
- If this is a GCC High / Azure US Government Cloud deployment then use this command instead:


```
Connect-AzAccount -EnvironmentName AzureUSGovernment
```

Then follow the prompts to sign in to Azure.

- b. Change directory to the folder into which you extracted the files from the Teams Connector ZIP.
- c. Run the following PowerShell commands to connect to Microsoft Graph:

```
# Set execution policy for the current PowerShell process, when prompted type A (Yes to All)
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# The Unblock-File cmdlet lets you run PowerShell script files that were downloaded from the internet.
# By default, these files are blocked to protect the computer from untrusted files.
Get-ChildItem -Recurse | Unblock-File

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph
```

- d. Run your variable initialization script to set the required prefix and region name variables.
- e. Run the following command to create the Teams Connector API app.

```
$teamsConnectorApiApp = New-MgApplication -DisplayName "${PxBaseConnName}-TeamsConn-${PxVmssRegion} Pexip Teams Connector API" -
SignInAudience "AzureADMyOrg"
```

Please allow one minute for this command to complete and propagate within Azure.

- f. Run the following commands to obtain the Teams Connector API app ID.

Note that if the **New-MgServicePrincipal** command fails, this means that you have not waited long enough for the previous command to complete.

```
$TeamsConnectorApiSp = New-MgServicePrincipal -AppId $TeamsConnectorApiApp.AppId -Tags {WindowsAzureActiveDirectoryIntegratedApp}
$TeamsConnectorApiApplicationId = $TeamsConnectorApiApp.AppId

Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host "### Teams Connector API App ID MUST be saved in the variables initialization script ###"
Write-Host
Write-Host "`$TeamsConnectorApiApplicationId = `"$($TeamsConnectorApiApplicationId)`""
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host
```

- g. When the command runs, it generates some output that lists the Teams Connector API App ID, similar to this:

```
### Teams Connector API App ID MUST be saved in the variables initialization script ###
$TeamsConnectorApiApplicationId = "36ee4c6c-0812-40a2-b820-b22ebd02bce4"
```

- h. Copy the output line that defines the API App ID and paste it into the variable initialization script, **replacing the existing variable definition** in the script that currently contains the API App ID for the main region used by the original variables script (the one you used as the basis for the variables script for this new region).

i.e. the script will initially contain:

```
$TeamsConnectorApiApplicationId = "<App ID for the existing main region>"
```

and it needs changing to the App ID created in step 2:

```
$TeamsConnectorApiApplicationId = "<App ID for the new region>"
```

(If you will be performing the rest of the deployment, in the same PowerShell session, there is no need to re-run the variable initialization script as the new \$TeamsConnectorApiApplicationId variable has been set in the previous steps.)

- i. If somebody else will be completing the deployment, ensure that the person who will perform all of the remaining installation steps has **Owner** permissions for the API app. See [Assigning Owner permissions for the API app](#) for more information.

Note that:

- This app does not have to be granted any permissions. It does not have access to any resources in the Azure AD tenant. It has no associated credentials.

5. Run the [redeploy script](#) to install the Teams Connector into the new region.

This should be the script that you produced after the initial installation or most recent upgrade. This script should already have the variables for the CVI App ID and password updated with the correct values for your deployment. As with the initial installation, we recommend running each group of commands step-by-step within PowerShell.

6. Update DNS with the new Teams Connector's name and IP address.

Note that you do **not** need to create a new CVI app or perform any additional app authorizations when installing additional Teams Connectors.

Pexip Management Node configuration

To configure Pexip Infinity to use the new Teams Connector:

1. If required, set up new Conferencing Nodes / locations that will use the new Teams Connector.
2. Ensure that the Conferencing Nodes have suitable certificates installed (see [Certificate and DNS examples for a Microsoft Teams integration](#)).
3. Configure a new **Microsoft Teams Connector** (Call Control > Microsoft Teams Connectors) where the address of the Teams Connector is the name specified in \$PxTeamsConnFqdn in your new initialization script for the new Azure region.
4. Assign the new Teams Connector to the locations (and thus Conferencing Nodes) where you want that new Teams Connector to be used (Platform > Locations).

5. Now that you have multiple Teams Connectors you must ensure that calls are routed via your desired Conferencing Nodes to the appropriate Teams Connector. This assumes that GeoDNS or your call control system is routing incoming calls to the appropriate Conferencing Nodes / locations.
 - When you have just one Teams Connector, all of the Call Routing Rules handling calls to Microsoft Teams are typically all configured with an **Outgoing location** set to the one location containing your Conferencing Nodes that communicate with your Teams Connector.
 - When you have two (or more) Teams Connectors, you will also have two (or more) locations, where each location typically contains the Conferencing Nodes that are local to each Teams Connector.

To ensure that your calls are routed via the most appropriate Conferencing Nodes and Teams Connectors, you must modify your existing Call Routing Rules and also create some new rules. Typically the best way to do this is to use the **Calls being handled in location** setting on each rule, so that you can route calls via the appropriate Teams Connector based on where the incoming call was originally received.

For example, if you previously only had a single location (called "Europe-DMZ") containing all of your Conferencing Nodes and that location was also configured as the **Outgoing location** on all of your rules, then all calls would have been received in — and routed out of — those nodes to your single Teams Connector (also most likely deployed in a European Azure region).

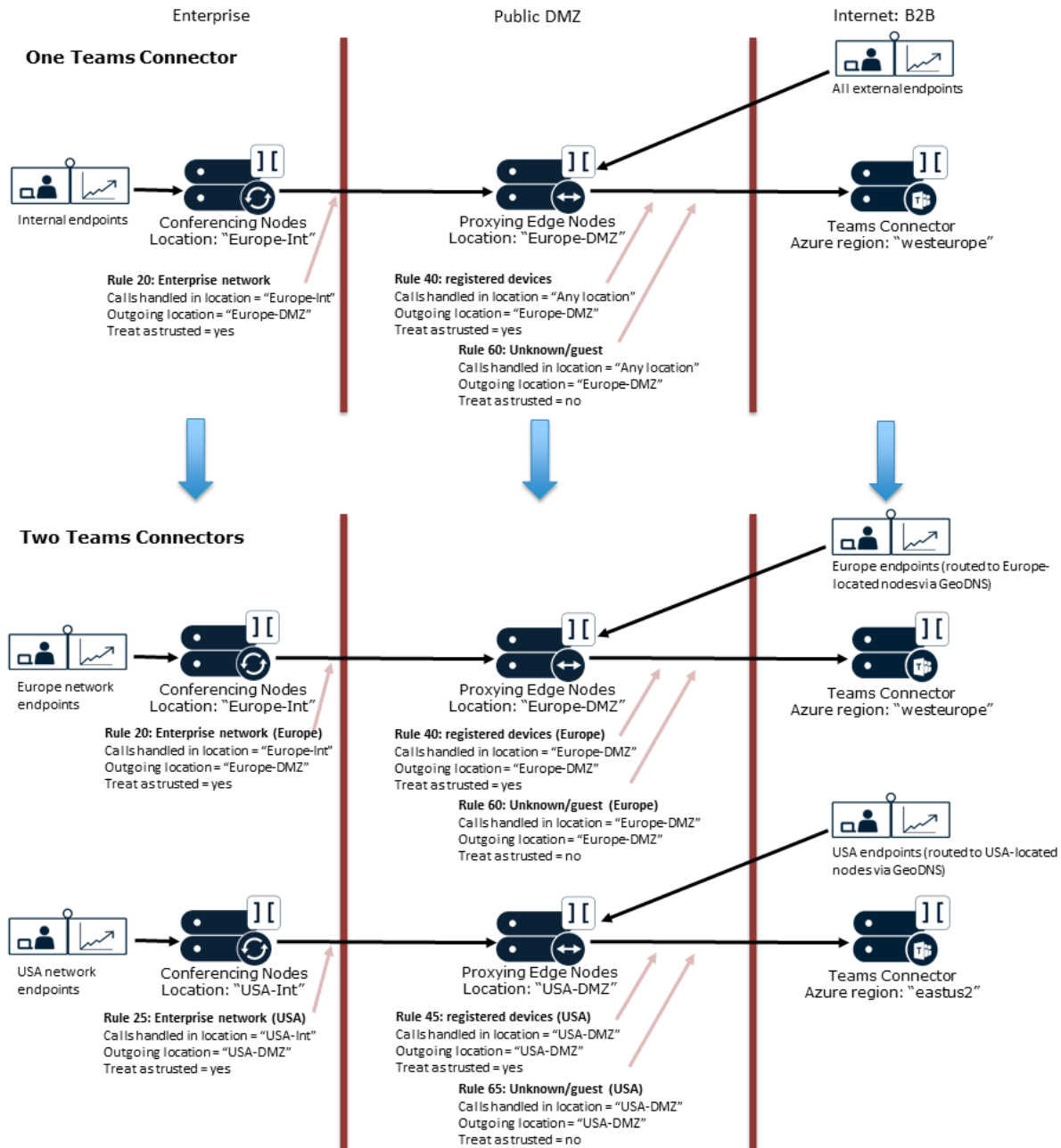
Let's assume that you now add a second location (called "USA-DMZ") containing new Conferencing Nodes and you want to route calls local to America to a new Teams Connector that is deployed in an American region, and keep your other calls routed via the Europe nodes and Europe Teams Connector. You now need to have another set of Call Routing Rules that are based on (i.e. copies of) your existing rules but where you:

- keep the same rule settings for protocols, alias regexes and trust settings across each pair of rules
- assign a **Priority** value for each new rule that keeps it next to the rule it was based upon
- specify the **Calls being handled in location** and **Outgoing location** fields as appropriate for each pair of rules, meaning that:
 - your original rule now has **Calls being handled in location** = "Europe-DMZ" and **Outgoing location** = "Europe-DMZ"
 - your new rule has **Calls being handled in location** = "USA-DMZ" and **Outgoing location** = "USA-DMZ"

Note that if you have many locations, you can specify some higher priority rules (lower numbers) with a specific location defined in **Calls being handled in location**, and have a lower priority rule with **Calls being handled in location** set to *Any location* to handle any calls that were not caught by the previous location-specific rules.

This means that Incoming calls will be routed via the Teams Connector that is local to the Conferencing Nodes that received the call.

This example diagram shows how your Call Routing Rules should evolve when moving from routing all calls via a single Teams Connector to deploying a second Teams Connector in another Azure region. Note that this example also includes locations/nodes in the enterprise internal network, in addition to the locations/nodes in the DMZ.



You do not have to change your existing Virtual Reception configuration — there is minimal performance overhead by always using a single location and Teams Connector to resolve Teams Conference IDs entered into a Virtual Reception.

Using certificate-based authentication for the Teams Connector CVI application

From version 33 you can use certificate-based authentication (CBA) to authenticate the Teams Connector CVI application towards MS Graph. In version 33, CBA is optional and the previous password-based authentication method is still the default mechanism.

- i** The CBA method will be the default and recommended mechanism in version 34. Password-based authentication will still be supported in version 34 but we plan to deprecate it in a future release, thus we recommend migrating to CBA as soon as practicable.

This topic explains how to use CBA instead of password-based authentication with version 33 of the Teams Connector. It covers:

- [Deploying a new \(first time\) Teams Connector with certificate-based authentication](#)
- [Upgrading or redeploying a Teams Connector that is already using CBA](#)
- [Migrating \(upgrading\) an existing Teams Connector using password-authentication to CBA](#)

Deploying a new (first time) Teams Connector with certificate-based authentication

You can follow most of the [standard steps](#) for installing the Teams Connector. However, there are some variations to the installation script and the post-deploy process as described below.

- i** Remember to run your standard variable initialization script first (the variable script is the same for either authentication method), before running this installation script.

The **installation script** for certificate-based authentication is provided below.

Note that:

- The script passes a `-ValidityInMonths` parameter to the `PexTeamsCviApplicationCertificateCredential` cmdlet to specify the validity period of the certificate. In this case it specifies `"-ValidityInMonths 24"` i.e. 2 years but you can specify your own period as required.
- The Teams Connector application will **stop working** if the CVI App certificate expires.

```
# Ensure the correct script and software combination is being used
try {$PxConnMajorVersion = (Get-Content .\version.json -ErrorAction Stop | Out-String | ConvertFrom-Json).major} catch {Write-Warning "Can't find version.json file. Make sure you run the installation script from the folder into which you extracted the files from the Teams Connector ZIP"}

if ($PxConnMajorVersion -ne "33"){Write-Warning "The Connector version (extracted ZIP files) and this deployment script version do not match. Connector version = $PxConnMajorVersion. Deployment script version = 33"}

# Connect to Microsoft Graph, Azure Resource Manager account and import Pexip CVI module
# Microsoft Graph commands
# Connect to Microsoft Graph
# If AAD/365 admin account is not the same as Azure Resource Manager admin account,
# the next section is to be run by the AAD admin.
#
# IMPORTANT: The output of IDs/credentials here must be saved as it will be required later

# Set execution policy for the current PowerShell process, when prompted type A (Yes to All)
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# The Unblock-File cmdlet lets you run PowerShell script files that were downloaded from the internet.
# By default, these files are blocked to protect the computer from untrusted files.
Get-ChildItem -Recurse | Unblock-File

# Connect to Azure with an authenticated account for use with Azure Resource Manager (in same window to reuse variables)
Connect-AzAccount

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1
```

```

# Connect to Graph
Connect-PexTeamsMsGraph

# Create Pexip CVI Application
# Create App
$App = New-PexTeamsCviApplication -AppDisplayName "$($PxBaseConnName)TeamsConn" -Confirm:$false
$AppId = $App.AppId

# Create App Certificate - the certificate validity period is defined via the -ValidityInMonths parameter passed to New-
PexTeamsCviApplicationCertificateCredential
# New-PexTeamsCviApplicationCertificateCredential cmdlet creates a self signed certificate and uploads it to Azure AD for use as a credential
for the CVI app
Write-Host "### Create and save CVI App certificate password using a password manager ###"
$AppCertificatePath = $App | New-PexTeamsCviApplicationCertificateCredential -ValidityInMonths 24
$AppCertificatePath = $AppCertificatePath.Trim()

Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host "### Save CVI App ID (use a password manager) ###"
Write-Host "### Save CVI App certificate ###"
Write-Host
Write-Host "`$AppId = `"$($AppId)`""
Write-Host "`$AppCertificatePath = `"$($AppCertificatePath)`""
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host
Write-Host

# Change context to the Pexip Subscription
Set-AzContext -SubscriptionId $PxSubscriptionId

# Azure Bot for the CVI AppID
# Create bot (must be globally unique, and only needs to be created once - in any of your regions)
Register-PexTeamsCviApplicationBot -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxBotResourceGroupName -BotName "$($PxBaseConnName)-
TeamsBot" -AppId $AppId -Confirm:$false -Tag $tags

# Deploy Pexip Teams Connector admin consent web page
$AdminConsentUrl = Publish-PexTeamsAdminConsentWebSite -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxBotResourceGroupName -
PxBaseConnName $PxBaseConnName -AppId $AppId -SourceAddressPrefixes $PxConsentSourceAddressPrefixes -Confirm:$false -Tag $tags

# Virtual Machine Scale Set (VMSS) creation
# Provide credentials to be used as local user/password for Pexip Teams Connector VMs
# Create a password (using the initialization script variables) for the Windows VM
$PxWinAdminSecurePassword = ConvertTo-SecureString -AsPlainText $PxWinAdminPassword -Force
$PxWinAdminCred = New-Object System.Management.Automation.PSCredential -ArgumentList $PxWinAdminUser, $PxWinAdminSecurePassword

# Optionally if you did not want to specify the password as a variable, use Get-Credential
# $PxWinAdminCred = Get-Credential

# Deploy the Teams Connector VMs
# this step can take up to 30 minutes to complete
.\create_vmss_deployment.ps1 -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxTeamsConnResourceGroupName -VmssName
"$($PxBaseConnName)$($PxVmssRegion)" -VMAdminCredential $PxWinAdminCred -PfxPath $PxPfxCertFileName -TeamsConnectorFqdn $PxTeamsConnFqdn -
PexipFqdns $PxNodeFqdns -InstanceCount $PxTeamsConnInstanceCount -AppId $AppId -AppCertificatePath $AppCertificatePath -
StaticResourcesResourceGroupName $PxTeamsConnStaticResourceGroupName -IncidentReporting $PxTeamsConnIncidentReporting -RdpSourceAddressPrefixes
$PxMgmtSrcAddrPrefixes -PexipSourceAddressPrefixes $PxNodesSourceAddressPrefixes -WupdScheduledInstallDay $PxWupdScheduledInstallDay -
WupdScheduledInstallTime $PxWupdScheduledInstallTime -WupdActiveHoursStart $PxWupdActiveHoursStart -WupdActiveHoursEnd $PxWupdActiveHoursEnd -
CustomerUsageAttribution $PxCustomerUsageAttribution -UseAzureHybridBenefit $PxUseAzureHybridBenefit -Tag $tags -TeamsConnectorApiApplicationId
$TeamsConnectorApiApplicationId -FunctionsDedicatedHostingPlan $FunctionsDedicatedHostingPlan -EventHubSourceAddressPrefixes
$EventHubSourceAddressPrefixes -VnetIntegration $VnetIntegration -PexipConfiguredConnectorFqdn $PexipConfiguredConnectorFqdn -PexipOutboundFqdn
$PexipOutboundFqdn -ExistingVNETResourceId $PxExistingVNETResourceId -UsePrivateRouting $PxUsePrivateRouting

# supply the PFX Teams Connector TLS certificate file password when prompted

# Please enter the password for the PFX Teams Connector TLS certificate '.\xxxxxxx.pfx': *****

# supply the PFX CVI app certificate file password when prompted

# Please enter the password for the CVI app PFX certificate '.\xxxxxxx.pfx': *****

# Generating the next steps summary (this assumes you are connected to Microsoft Graph and with an authenticated account for use with Azure
Resource Manager)
#
# Setting subscription

```



```

Set-AzContext -SubscriptionId $PxSubscriptionId

# Getting IP configurations
if ($PxUsePrivateRouting) {
    $LB = Get-AzLoadBalancer -ResourceGroupName $PxTeamsConnResourceGroupName
    $ExtLB = $LB | Where-Object { $_.Name.EndsWith("-LB") }
    $IntLB = $LB | Where-Object { $_.Name.EndsWith("-INTLB") }
    $LBExtIPID = $ExtLB.FrontendIpConfigurations[0].PublicIpAddress.id
    $LBIntIP = $IntLB.FrontendIpConfigurations[0].PrivateIpAddress
    $PublicIPAddresses = Get-AzPublicIpAddress -ResourceGroupName $PxTeamsConnStaticResourceGroupName
    $ConnectorPublicIP = $PublicIPAddresses | Where-Object Id -eq $LBExtIPID
    $publicIpAddress = $ConnectorPublicIP[0].IpAddress
    $privateIpAddress = $LBIntIP
} else {
    $LB = (Get-AzLoadBalancer -ResourceGroupName $PxTeamsConnResourceGroupName)[0]
    $LBPublicIPID = $LB.FrontendIpConfigurations[0].PublicIpAddress.id
    $PublicIPAddresses = Get-AzPublicIpAddress -ResourceGroupName $PxTeamsConnStaticResourceGroupName
    $ConnectorPublicIP = $PublicIPAddresses | Where-Object Id -eq $LBPublicIPID
    $publicIpAddress = $ConnectorPublicIP[0].IpAddress
}

# Getting connection string for the newly deployed Event hub
$eventHub = (Get-AzResource -ResourceGroupName $PxTeamsConnStaticResourceGroupName -ResourceType Microsoft.EventHub/namespaces)[0]
$eventHubKey = Get-AzEventHubKey -Name "pexip_teams_connector_access" -NamespaceName $eventHub.Name -ResourceGroupName $eventHub.ResourceGroupName

# Printing next steps
Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host "When the Teams Connector is deployed, you have to create a DNS A record for your hostname,"
Write-Host "then the Office 365 admin must consent for the CVI App Id to join Teams Meetings"
Write-Host
Write-Host "1) Set up a public DNS A record for $($PxTeamsConnFqdn) pointing to the Public IP of "
Write-Host "    the load balancer $($publicIpAddress)"
Write-Host
Write-Host "2) Give consent to the CVI app. Go to: $AdminConsentUrl"
Write-Host
Write-Host "    If Management Consent Source Address prefixes are defined, the administrator"
Write-Host "    doing consent must come from one of these addresses (or subnets)."
Write-Host "    Consent address prefixes: $($PxConsentSourceAddressPrefixes)"
Write-Host
Write-Host "3) Update the Management Node setting 'Azure Event Hub connection string' for $($PxTeamsConnFqdn) to:"
Write-Host "    $($eventHubKey.PrimaryConnectionString)"
Write-Host
if ($PxUsePrivateRouting) {
    Write-Host "4) Set up a private DNS A record for $($PxTeamsConnFqdn) pointing to the (private) frontend IP of "
    Write-Host "    the internal load balancer $($privateIpAddress)"
    Write-Host "    See: https://docs.pexip.com/admin/teams\_routing.htm#enabling"
}
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host

```

```

# This script only applies to GCC High / Azure US Government Cloud deployments

# Ensure the correct script and software combination is being used
try {$PxConnMajorVersion = (Get-Content .\version.json -ErrorAction Stop | Out-String | ConvertFrom-Json).major} catch {Write-Warning "Can't find version.json file. Make sure you run the installation script from the folder into which you extracted the files from the Teams Connector ZIP"}

if ($PxConnMajorVersion -ne "33"){Write-Warning "The Connector version (extracted ZIP files) and this deployment script version do not match. Connector version = $PxConnMajorVersion. Deployment script version = 33"}

# Set VmImage variable to hold the CIS STIG image properties - STIG image is optional but typical
# In a later step in this script you can choose not to use the STIG image
$VmImage = @{
    "sku" = "cis-win-2019-stig"
    "offer" = "cis-win-2019-stig"
    "publisher" = "center-for-internet-security-inc"
}

```

```

"version" = "latest"}

# Connect to Microsoft Graph, Azure Resource Manager account and import Pexip CVI module
# Microsoft Graph commands
# Connect to Microsoft Graph
# If AAD/365 admin account is not the same as Azure Resource Manager admin account,
# the next section is to be run by the AAD admin.
#
# IMPORTANT: The output of IDs/credentials here must be saved as it will be required later

# Set execution policy for the current PowerShell process, when prompted type A (Yes to All)
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# The Unblock-File cmdlet lets you run PowerShell script files that were downloaded from the internet.
# By default, these files are blocked to protect the computer from untrusted files.
Get-ChildItem -Recurse | Unblock-File

# Connect to Azure USGovernment with an authenticated account for use with Azure Resource Manager (in same window to reuse variables)
Connect-AzAccount -EnvironmentName AzureUSGovernment

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph

# Create Pexip CVI Application
# Create App
$App = New-PexTeamsCviApplication -AppDisplayName "$($PxBaseConnName)TeamsConn" -Confirm:$false
$AppId = $App.AppId

# Create App Certificate - the certificate validity period is defined via the -ValidityInMonths parameter passed to New-
PexTeamsCviApplicationCertificateCredential
# New-PexTeamsCviApplicationCertificateCredential cmdlet creates a self signed certificate and uploads it to Azure AD for use as a credential
for the CVI app
Write-Host "### Create and save CVI App certificate password using a password manager ###"
$AppCertificatePath = $App | New-PexTeamsCviApplicationCertificateCredential -ValidityInMonths 24
$AppCertificatePath = $AppCertificatePath.Trim()

Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host "### Save CVI App ID (use a password manager) ###"
Write-Host "### Save CVI App certificate ###"
Write-Host
Write-Host "`$AppId = `"$($AppId)`""
Write-Host "`$AppCertificatePath = `"$($AppCertificatePath)`""
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host

# Change context to the Pexip Subscription
Set-AzContext -SubscriptionId $PxSubscriptionId

# Azure Bot for the CVI AppID
# Create bot (must be globally unique, and only needs to be created once - in any of your regions)
Register-PexTeamsCviApplicationBot -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxBotResourceGroupName -BotName "$($PxBaseConnName)-
TeamsBot" -AppId $AppId -Confirm:$false -Tag $tags -TeamsEnvironmentName TeamsGCCHigh

# Deploy Pexip Teams Connector admin consent web page
$AdminConsentUrl = Publish-PexTeamsAdminConsentWebSite -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxBotResourceGroupName -
PxBaseConnName $PxBaseConnName -AppId $AppId -SourceAddressPrefixes $PxConsentSourceAddressPrefixes -Confirm:$false -Tag $tags

# Virtual Machine Scale Set (VMSS) creation
# Provide credentials to be used as local user/password for Pexip Teams Connector VMs
# Create a password (using the initialization script variables) for the Windows VM
$PxWinAdminSecurePassword = ConvertTo-SecureString -AsPlainText $PxWinAdminPassword -Force
$PxWinAdminCred = New-Object System.Management.Automation.PSCredential -ArgumentList $PxWinAdminUser,$PxWinAdminSecurePassword

# Optionally if you did not want to specify the password as a variable, use Get-Credential
# $PxWinAdminCred = Get-Credential

# Deploy the Teams Connector VMs

```

```
# this step can take up to 30 minutes to complete
# if you are not using a STIG image then remove the following parameter from this command: -VmImage $VmImage
.\create_vmss_deployment.ps1 -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxTeamsConnResourceGroupName -VmssName
"$($PxBaseConnName)$($PxVmssRegion)" -VmAdminCredential $PxWinAdminCred -PfxPath $PxPfxCertFileName -TeamsConnectorFqdn $PxTeamsConnFqdn -
PexipFqdns $PxNodeFqdns -InstanceCount $PxTeamsConnInstanceCount -AppId $AppId -AppCertificatePath $AppCertificatePath -
StaticResourcesResourceGroupName $PxTeamsConnStaticResourceGroupName -IncidentReporting $PxTeamsConnIncidentReporting -RdpSourceAddressPrefixes
$PxMgmtSrcAddrPrefixes -PexipSourceAddressPrefixes $PxNodesSourceAddressPrefixes -WupdScheduledInstallDay $PxWupdScheduledInstallDay -
WupdScheduledInstallTime $PxWupdScheduledInstallTime -WupdActiveHoursStart $PxWupdActiveHoursStart -WupdActiveHoursEnd $PxWupdActiveHoursEnd -
CustomerUsageAttribution $PxCustomerUsageAttribution -UseAzureHybridBenefit $PxUseAzureHybridBenefit -Tag $Tags -TeamsConnectorApiApplicationId
$TeamsConnectorApiApplicationId -FunctionsDedicatedHostingPlan $FunctionsDedicatedHostingPlan -EventHubSourceAddressPrefixes
$EventHubSourceAddressPrefixes -VnetIntegration $VnetIntegration -VmImage $VmImage -TeamsEnvironmentName TeamsGCCHigh -
PexipConfiguredConnectorFqdn $PexipConfiguredConnectorFqdn -PexipOutboundFqdn $PexipOutboundFqdn -ExistingVNETResourceId
$PxExistingVNETResourceId -UsePrivateRouting $PxUsePrivateRouting

# supply the PFX Teams Connector TLS certificate file password when prompted

# Please enter the password for the PFX Teams Connector TLS certificate '.\xxxxxxx.pfx': *****

# supply the PFX CVI app certificate file password when prompted

# Please enter the password for the CVI app PFX certificate '.\xxxxxxx.pfx': *****

# Generating the next steps summary (this assumes you are connected to Microsoft Graph and with an authenticated account for use with Azure
Resource Manager)
#
# Setting subscription
Set-AzContext -SubscriptionId $PxSubscriptionId

# Getting IP configurations
if ($PxUsePrivateRouting) {
    $LB = Get-AzLoadBalancer -ResourceGroupName $PxTeamsConnResourceGroupName
    $ExtLB = $LB | Where-Object { $_.Name.EndsWith("-LB") }
    $IntLB = $LB | Where-Object { $_.Name.EndsWith("-INTLB") }
    $LBExtIPID = $ExtLB.FrontendIpConfigurations[0].PublicIpAddress.id
    $LBIntIP = $IntLB.FrontendIpConfigurations[0].PrivateIpAddress
    $PublicIPAddresses = Get-AzPublicIpAddress -ResourceGroupName $PxTeamsConnStaticResourceGroupName
    $ConnectorPublicIP = $PublicIPAddresses | Where-Object Id -eq $LBExtIPID
    $publicIpAddress = $ConnectorPublicIP[0].IpAddress
    $privateIpAddress = $LBIntIP
} else {
    $LB = (Get-AzLoadBalancer -ResourceGroupName $PxTeamsConnResourceGroupName)[0]
    $LBPublicIPID = $LB.FrontendIpConfigurations[0].PublicIpAddress.id
    $PublicIPAddresses = Get-AzPublicIpAddress -ResourceGroupName $PxTeamsConnStaticResourceGroupName
    $ConnectorPublicIP = $PublicIPAddresses | Where-Object Id -eq $LBPublicIPID
    $publicIpAddress = $ConnectorPublicIP[0].IpAddress
}

# Getting connection string for the newly deployed Event hub
$eventHub = (Get-AzResource -ResourceGroupName $PxTeamsConnStaticResourceGroupName -ResourceType Microsoft.EventHub/namespaces)[0]
$eventHubKey = Get-AzEventHubKey -Name "pexip_teams_connector_access" -NamespaceName $eventHub.Name -ResourceGroupName
$eventHub.ResourceGroupName

# Printing next steps
Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host "When the Teams Connector is deployed, you have to create a DNS A record for your hostname,"
Write-Host "then the Office 365 admin must consent for the CVI App Id to join Teams Meetings"
Write-Host
Write-Host "1) Set up a public DNS A record for $($PxTeamsConnFqdn) pointing to the Public IP of "
Write-Host "    the load balancer $($publicIpAddress)"
Write-Host
Write-Host "2) Give consent to the CVI app. Go to: $AdminConsentUrl"
Write-Host
Write-Host "    If Management Consent Source Address prefixes are defined, the administrator"
Write-Host "    doing consent must come from one of these addresses (or subnets)."
```

```
}
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host
```

After deploying the (first) Teams Connector

1. When the script ran, it generated some output that listed the CVI App ID and certificate path:
2. Store the items as instructed:
 - a. Store the CVI App ID and CVI App certificate password in a password manager. These variables are needed to redeploy the Teams Connector.
 - b. Store the CVI App certificate PFX file in a safe location.

This means that if you need to run the redeploy script, you will not rerun the commands that imported the PowerShell module and created the app. Instead you will set the variables to the ID and certificate of the CVI app that you created the first time.

Upgrading or redeploying a Teams Connector that is already using CBA

You can follow most of the [standard steps](#) for upgrading and redeploying the **Teams Connector**. However, there are some variations to the process and scripts as described below.

Check Teams Connector software, retrieve your original scripts, and check your Azure environment

In this section, step 4 is new and step 7 (previously step 6) has been updated to refer to assigning the `$AppId` and `$AppCertificatePath` variables.

1. Download the latest relevant version of the **Teams Connector ZIP** file (Pexip_Infinity_Connector_For_Microsoft_Teams_v33_<build>.zip) from the [Pexip download page](#).
Ensure that the Teams Connector version you download is the same version as your Pexip Infinity deployment (including minor/ "dot" releases).
2. Extract the files to a folder on a local drive on your administrator PC.
3. Add your [PFX Teams Connector TLS certificate](#) to this folder.
4. **Add your PFX certificate for the Teams Connector CVI app to this folder.**
5. Retrieve your saved copies of the initialization and redeploy scripts. You should have created and stored your version of these scripts after you completed your initial installation of your first Teams Connector.

If you are migrating (upgrading) an existing Teams Connector using password-authentication to CBA, use the initialization and redeploy scripts contained on this page.

6. Check your saved copy of the initialization script that sets the environmental variables:
 - If you are redeploying and need to change any of the previous configuration you should also adjust your initialization script as required.
 - If you have Teams Connectors in many regions, ensure that you have the correct versions of the initialization scripts that set the regional variables to the appropriate values.
7. **Check your saved copy of the redeploy script:**
 - The CVI App ID and CVI App certificate password should have been stored in a password manager.
Use the stored CVI App ID value to update the first of the two existing lines in the redeploy script (further below) that say:

```
$AppId = ""
$AppCertificatePath = ".\your_cvi_app_certificate.pfx"
```

 and set `$AppCertificatePath` to refer to the file name of the certificate that you retrieved in step 4.
You'll be prompted for the CVI App certificate password later.
This means that when you run the redeploy script, you will reuse the CVI app and CVI app certificate that you created the first time.
 - You should use the redeploy script in all scenarios except for when deploying a Teams Connector for the very first time for your Azure subscription i.e. you should use the redeploy script when upgrading, redeploying, or deploying a new Teams Connector in another region.

- You only need one version of this script — you can use the same redeploy script in every region if you have multiple Teams Connectors.
8. As usual, remove and then recreate the dynamic resource group and ensure appropriate roles are assigned to the resource groups as described [here](#).

Redeploy the Teams Connector

In this section you must use the new version of the redeploy script as provided below, which uses your certificate to authenticate the Teams Connector CVI application towards MS Graph.

This is the redeploy script. It is a variation on the installation script that only performs the necessary commands to redeploy the Teams Connector. As with the initial installation, we recommend running each group of commands step-by-step within PowerShell.

```
# Ensure the correct script and software combination is being used
try {$PxConnMajorVersion = (Get-Content .\version.json -ErrorAction Stop | Out-String | ConvertFrom-Json).major} catch {Write-Warning "Can't find version.json file. Make sure you run the installation script from the folder into which you extracted the files from the Teams Connector ZIP"}

if ($PxConnMajorVersion -ne "33"){Write-Warning "The Connector version (extracted ZIP files) and this deployment script version do not match. Connector version = $PxConnMajorVersion. Deployment script version = 33"}

# Connect to PowerShell Azure Resource Manager account
# Set execution policy for the current PowerShell process, when prompted type A (Yes to All)
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# The Unblock-File cmdlet lets you run PowerShell script files that were downloaded from the internet.
# By default, these files are blocked to protect the computer from untrusted files.
Get-ChildItem -Recurse | Unblock-File

# Connect to Azure with an authenticated account for use with Azure Resource Manager (in same window to reuse variables)
Connect-AzAccount

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph

# Before running the following commands, update the following 2 lines/variables with the CVI App ID and
# the path to the CVI App certificate file that were output when the original installation script was run
# You'll be prompted for the CVI App certificate password later.
$AppId = ""
$AppCertificatePath = ".\your_cvi_app_certificate.pfx"

# Change context to the Pexip Subscription
Set-AzContext -SubscriptionId $PxSubscriptionId

# Virtual Machine Scale Set (VMSS) creation
# Provide credentials to be used as local user/password for Pexip Teams Connector VMs
# Create a password (using the initialization script variables) for the Windows VM
$PxWinAdminSecurePassword = ConvertTo-SecureString -AsPlainText $PxWinAdminPassword -Force
$PxWinAdminCred = New-Object System.Management.Automation.PSCredential -ArgumentList $PxWinAdminUser, $PxWinAdminSecurePassword

# Optionally if you did not want to specify the password as a variable, use Get-Credential
# $PxWinAdminCred = Get-Credential

# Deploy the Teams Connector VMs
# this step can take up to 30 minutes to complete
.\create_vmss_deployment.ps1 -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxTeamsConnResourceGroupName -VmssName
"$($PxBaseConnName)$($PxVmssRegion)" -VMAdminCredential $PxWinAdminCred -PfxPath $PxPfxCertFileName -TeamsConnectorFqdn $PxTeamsConnFqdn -
PexipFqdns $PxNodeFqdns -InstanceCount $PxTeamsConnInstanceCount -AppId $AppId -AppCertificatePath $AppCertificatePath -
StaticResourcesResourceGroupName $PxTeamsConnStaticResourceGroupName -IncidentReporting $PxTeamsConnIncidentReporting -RdpSourceAddressPrefixes
$PxMgmtSrcAddrPrefixes -PexipSourceAddressPrefixes $PxNodesSourceAddressPrefixes -WupdScheduledInstallDay $PxWupdScheduledInstallDay -
WupdScheduledInstallTime $PxWupdScheduledInstallTime -WupdActiveHoursStart $PxWupdActiveHoursStart -WupdActiveHoursEnd $PxWupdActiveHoursEnd -
CustomerUsageAttribution $PxCustomerUsageAttribution -UseAzureHybridBenefit $PxUseAzureHybridBenefit -Tag $tags -TeamsConnectorApiApplicationId
$TeamsConnectorApiApplicationId -FunctionsDedicatedHostingPlan $FunctionsDedicatedHostingPlan -EventHubSourceAddressPrefixes
$EventHubSourceAddressPrefixes -VnetIntegration $VnetIntegration -PexipConfiguredConnectorFqdn $PexipConfiguredConnectorFqdn -PexipOutboundFqdn
$PexipOutboundFqdn -ExistingVNETResourceId $PxExistingVNETResourceId -UsePrivateRouting $PxUsePrivateRouting

# supply the PFX certificate file password when prompted

# Please enter the password for the PFX certificate '.\xxxxxxx.pfx': *****

# supply the PFX CVI app certificate file password when prompted
```

```
# Please enter the password for the CVI app PFX certificate '.\xxxxxxx.pfx': *****

# Printing finished message
Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host " All steps completed."
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host
```

```
# This script only applies to GCC High / Azure US Government Cloud deployments

# Ensure the correct script and software combination is being used
try {$PxConnMajorVersion = (Get-Content .\version.json -ErrorAction Stop | Out-String | ConvertFrom-Json).major} catch {Write-Warning "Can't find version.json file. Make sure you run the installation script from the folder into which you extracted the files from the Teams Connector ZIP"}

if ($PxConnMajorVersion -ne "33"){Write-Warning "The Connector version (extracted ZIP files) and this deployment script version do not match. Connector version = $PxConnMajorVersion. Deployment script version = 33"}

# Set VmImage variable to hold the CIS STIG image properties - STIG image is optional but typical
# In a later step in this script you can choose not to use the STIG image
$VmImage = @{
    "sku"      = "cis-win-2019-stig"
    "offer"    = "cis-win-2019-stig"
    "publisher" = "center-for-internet-security-inc"
    "version"  = "latest"}

# Connect to PowerShell Azure Resource Manager account
# Set execution policy for the current PowerShell process, when prompted type A (Yes to All)
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# The Unblock-File cmdlet lets you run PowerShell script files that were downloaded from the internet.
# By default, these files are blocked to protect the computer from untrusted files.
Get-ChildItem -Recurse | Unblock-File

# Connect to Azure USGovernment with an authenticated account for use with Azure Resource Manager (in same window to reuse variables)
Connect-AzAccount -EnvironmentName AzureUSGovernment

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph

# Before running the following commands, update the following 2 lines/variables with the CVI App ID and
# the path to the CVI App certificate file that were output when the original installation script was run
# You'll be prompted for the CVI App certificate password later.
$AppId = ""
$AppCertificatePath = ".\your_cvi_app_certificate.pfx"

# Change context to the Pexip Subscription
Set-AzContext -SubscriptionId $PxSubscriptionId

# Virtual Machine Scale Set (VMSS) creation
# Provide credentials to be used as local user/password for Pexip Teams Connector VMs
# Create a password (using the initialization script variables) for the Windows VM
$PxWinAdminSecurePassword = ConvertTo-SecureString $PxWinAdminPassword -Force
$PxWinAdminCred = New-Object System.Management.Automation.PSCredential -ArgumentList $PxWinAdminUser,$PxWinAdminSecurePassword

# Optionally if you did not want to specify the password as a variable, use Get-Credential
# $PxWinAdminCred = Get-Credential

# Deploy the Teams Connector VMs
# this step can take up to 30 minutes to complete
# if you are not using a STIG image then remove the following parameter from this command: -VmImage $VmImage
```

```
.\create_vmss_deployment.ps1 -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxTeamsConnResourceGroupName -VmssName
"$($PxBaseConnName)$($PxVmssRegion)" -VMAdminCredential $PxWinAdminCred -PfxPath $PxPfxCertFileName -TeamsConnectorFqdn $PxTeamsConnFqdn -
PexipFqdns $PxNodeFqdns -instanceCount $PxTeamsConnInstanceCount -AppId $AppId -AppCertificatePath $AppCertificatePath -
StaticResourcesResourceGroupName $PxTeamsConnStaticResourceGroupName -IncidentReporting $PxTeamsConnIncidentReporting -RdpSourceAddressPrefixes
$PxMgmtSrcAddrPrefixes -PexipSourceAddressPrefixes $PxNodesSourceAddressPrefixes -WupdScheduledInstallDay $PxWupdScheduledInstallDay -
WupdScheduledInstallTime $PxWupdScheduledInstallTime -WupdActiveHoursStart $PxWupdActiveHoursStart -WupdActiveHoursEnd $PxWupdActiveHoursEnd -
CustomerUsageAttribution $PxCustomerUsageAttribution -UseAzureHybridBenefit $PxUseAzureHybridBenefit -Tag $tags -TeamsConnectorApiApplicationId
$TeamsConnectorApiApplicationId -FunctionsDedicatedHostingPlan $FunctionsDedicatedHostingPlan -EventHubSourceAddressPrefixes
$EventHubSourceAddressPrefixes -VnetIntegration $VnetIntegration -VmImage $VmImage -TeamsEnvironmentName TeamsGCCHigh -
PexipConfiguredConnectorFqdn $PexipConfiguredConnectorFqdn -PexipOutboundFqdn $PexipOutboundFqdn -ExistingVNETResourceId
$PxExistingVNETResourceId -UsePrivateRouting $PxUsePrivateRouting

# supply the PFX certificate file password when prompted

# Please enter the password for the PFX certificate '.\xxxxxxx.pfx': *****

# supply the PFX CVI app certificate file password when prompted

# Please enter the password for the CVI app PFX certificate '.\xxxxxxx.pfx': *****

# Printing finished message
Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host " All steps completed."
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host
```

Migrating (upgrading) an existing Teams Connector using password-authentication to CBA

If you already have a Teams Connector deployed with password-based authentication, you can switch to a certificate-based authentication model.

To switch to certificate-based authentication:

1. Download the latest relevant version of the **Teams Connector ZIP** file (Pexip_Infinity_Connector_For_Microsoft_Teams_v33_<build>.zip) from the [Pexip download page](#).
2. Extract the files to a folder on a local drive on your administrator PC.
3. Start a PowerShell session as Administrator.
4. After launching PowerShell you must change directory to the folder into which you extracted the files from the Teams Connector ZIP.
5. Assign the \$AppId variable with your CVI App ID (as contained within your existing redeploy script).
6. Run the following commands:

```
# Ensure the correct script and software combination is being used
try {$PxConnMajorVersion = (Get-Content .\version.json -ErrorAction Stop | Out-String | ConvertFrom-Json).major} catch {Write-
Warning "Can't find version.json file. Make sure you run the installation script from the folder into which you extracted the files from
the Teams Connector ZIP"}

if ($PxConnMajorVersion -ne "33"){Write-Warning "The Connector version (extracted ZIP files) and this deployment script version do not
match. Connector version = $PxConnMajorVersion. Deployment script version = 33"}
Connect-AzAccount

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph

$AppCertificatePath = New-PexTeamsCviApplicationCertificateCredential -AppId $AppId -ValidityInMonths 24
$AppCertificatePath = $AppCertificatePath.Trim()
# Please enter the password for the CVI app PFX certificate: *****
```

```
# Verifying - Enter the password for the CVI app PFX certificate: *****
# Your PFX certificate including the private key has been stored at '{absolute path}'
# The certificate is valid from '{notBefore}' to '{notAfter}' (UTC)
# CVI application certificate credential 'pexip-cvi-app-certificate-[{date}]' has been successfully uploaded to Azure AD
```

Enter the password for the CVI app PFX certificate when prompted. We recommend that you create and save the CVI App PFX certificate password using a password manager.

Note that:

- The script passes a `-ValidityInMonths` parameter to the **PexTeamsCviApplicationCertificateCredential** cmdlet to specify the validity period of the certificate. In this case it specifies "`-ValidityInMonths 24`" i.e. 2 years but you can specify your own period as required.
 - The Teams Connector application will **stop working** if the CVI App certificate expires.
7. Store the CVI App certificate PFX file in a safe location.
 8. Now you can follow the [redeploy instructions](#) contained here.

Using private routing with the Teams Connector

Private routing enables traffic between Pexip Infinity and a Teams Connector to go over a private/internal network instead of the public internet.

Standard Teams Connector deployments require public-facing Proxying Edge Nodes to establish connectivity between Pexip Infinity and the Teams Connector in Azure (this still applies even if the Pexip Infinity deployment resides solely in Azure). In addition to increased privacy and security, enabling private routing between the Pexip Infinity and Teams Connector deployments also has the benefit of removing the dependency on proxying nodes — instead you can set up virtual network peering between your Pexip Infinity platform and the Teams Connector.

 Using private routing with the Teams Connector is a technology preview feature.

Deployment environments and inter-network connectivity options

Your Pexip Infinity platform can be hosted in any of our supported deployment environments. Your choice of environment influences how you can set up links between your Pexip Infinity network and the Teams Connector network in Azure:

- If your Pexip Infinity deployment is also in Azure you can use VNET peering. Hosting everything in Azure may provide the best security/isolation, and Azure Virtual Network Peering should also offer the lowest latency between the two platforms if your Pexip Infinity deployment resides entirely within the same Azure region as your Teams Connector (although you must also consider the latency between your endpoints and Pexip Infinity).
- If your Pexip Infinity deployment is hosted in another cloud provider such as AWS or GCP, you will need to use some form of cross-cloud interconnect solution, such as a VPN Gateway to set up a VPN connection between the two environments.
- If you have an on-premises Pexip Infinity deployment you can use a site-to-site VPN between the on-premises network and Azure, or use a solution such as Azure ExpressRoute.

Requirements

Here are the requirements for setting up private routing:

- The link that connects the Teams Connector VNET to the other private network where Pexip Infinity is deployed (by whichever method — VNET peering within Azure, a site-to-site VPN, ExpressRoute to an on-premises network, or a cross-cloud interconnect) must be configured to:
 - Route the IP address range of the Teams Connector VNET from the private network to the Teams Connector VNET, and
 - Route the IP address range of Pexip Infinity from the Teams Connector VNET to the private network.

To achieve this, the following requirements must be met:

- Unique addressability within the entire private network: when you connect the VNET of a Teams Connector installation to another private network, you need to ensure that the address space of the Teams Connector VNET and its subnets is unique, i.e. it cannot overlap with any existing VLAN, subnet or other network resource in the connected private network. This ensures the link can route requests to the correct destination.
- Two-way routability between Pexip Infinity and the Teams Connector: the network must be able to route requests from Pexip Infinity to the Teams Connector deployment and requests from the Teams Connector deployment to Pexip Infinity over the link that connects the existing private network to the Teams Connector VNET.
- A split-DNS setup is required as Pexip Infinity must be able to resolve the FQDN of the Teams Connector load balancer to its private IP address, but Microsoft Teams must also be able to resolve the FQDN to its public IP address. Thus you must have the relevant A records both in internal DNS and public DNS. You can achieve the necessary internal DNS requirements by configuring a DNS A record in a private DNS zone in Azure, if required.
- Note that the TLS certificate installed on the Teams Connectors still must be a publicly-signed certificate.

Enabling private routing

The following instructions explain how to set up and enable private routing.

Note that steps 1 to 4 can be performed in advance of deploying the Teams Connector.

1. Create a Teams Connector VNET:

- You must use the `create_vnet_deployment.ps1` script from the Teams Connector ZIP file. If you create the VNET manually it will not contain the prerequisites required by the Teams Connector deployment.
- This VNET cannot overlap with the IP address space of the Pexip Infinity VNET.
 - The `create_vnet_deployment.ps1` script creates a /23 VNET by default. The range of this VNET as well as its subnets can be adjusted so that they do not overlap with the Pexip Infinity VNET.
 - For example, if you deployed the Pexip Infinity VNET/subnet with range of 10.0.0.0/24, you could use a range of 10.20.0.0/23 for the Teams Connector VNET and with subnets with ranges 10.20.0.0/24 and 10.20.1.0/28.
 - If you use a smaller subnet size, such as /28, for a small Teams Connector deployment, note that Azure uses 5 of the addresses in the subnet range (see [this article](#) for details).
- The VNET must be in the same subscription as your Teams Connector.
- We recommend that you assign the `$resourceGroupName` variable in the following example script to use the static resource group for the VNET. It will be created if it does not already exist

Using the example subnet ranges described above, you would run the `create_vnet_deployment.ps1` script as:

```
# Name of the resource group to use. We recommend using the static resource group i.e. $PxTeamsConnStaticResourceGroupName
# in the variable script. For example "pexip-TeamsConn-eu-static-RG"
# The resource group will be created if it does not already exist
$resourceGroupName = ""

# Azure Region (must be a supported region) in lower case ($PxAzureLocation in the variable script)
# For example $location = "westeurope"
$location = ""

# Optional tags (name-value pairs) to apply to Azure resources and resource groups ($tags in the variable script)
# For example $tags= @{ "ResourceOwner"="user@domain"; "CostCenter"="Video Services"; }
$tags = @{}

Connect-AzAccount

$exists = Get-AzResourceGroup -Name $resourceGroupName -Location $PxAzureLocation -ErrorAction SilentlyContinue
if (!$exists) {
    New-AzResourceGroup -Name $resourceGroupName -Location $PxAzureLocation -Tag $tags
} else {
    Write-Host "The resource group $($exists.ResourceGroupName) already exists."
}

# Make sure to call this script from the expanded TeamsConnector ZIP folder

./create_vnet_deployment.ps1 -ResourceGroupName $resourceGroupName -VnetName "pexip-teams-connector-VNET" -VnetAddressPrefixes
"10.20.0.0/23" -VmssSubnetAddressPrefix "10.20.0.0/24" -FuncSubnetAddressPrefix "10.20.1.0/28" -Tags $tags

# Script output:
# Deploying VNET
# Deployed VNET: pexip-teams-connector-VNET
# Deployed VNET resource ID: {VNET resource ID}
```

2. Set up VNET peering between the Pexip Infinity VNET and the Teams Connector VNET.

The specific details of how you set up peering to the Teams Connector VNET depends on your Pexip Infinity environment — see the [Deployment environments and inter-network connectivity options](#) section above for more information. Also, if relevant, see [Setting up peering where the Pexip Infinity platform is in Azure](#).

3. If you are also following (or planning to follow) a [blue-green deployment strategy](#), you should repeat steps 1 and 2 to set up a second VNET in the static resource group and configure VNET peering for it:

- Use the name of the other e.g. "green" static resource group for the `$resourceGroupName` variable.
- Ensure that the second VNET's address space **does not overlap** with the first VNET or the Pexip Infinity subnet range.

We recommend setting up both VNETs as part of the initial setup so that you can plan all of your subnet assignments and simplify the first upgrade process as you will have the alternative VNET and peering already in place.

4. Set the variables in the variable initialization script:

\$PxExistingVNETResourceId	<p>You must set this variable to the resource ID of the VNET created in step 1.</p> <p>Use the output of the <code>create_vnet_deployment.ps1</code> script — line "Deployed VNET resource ID: {VNET resource ID}".</p> <p>The resource ID can also be retrieved from Azure Portal > VNET resource > JSON view > Resource ID.</p>
\$PxUsePrivateRouting	<p>Set this to \$true to enable private routing.</p> <p>The deployment process will create an additional internal load balancer, allowing traffic to be routed between Pexip Infinity and the Teams Connector through Microsoft's private network only.</p>

Note that these variables are passed as parameters (`-ExistingVNETResourceId` and `-UsePrivateRouting`) to `create_vmss_deployment.ps1` in the installation and redeploy scripts.

- You can configure the deployed Teams Connector in the Pexip Infinity Management Node according to the [main documentation](#).
- After the Teams Connector deployment has finished you must configure an additional DNS record:
 - In addition to creating a **public** A record pointing to the external IP of the Azure load balancer, you must also create an **internal** DNS record pointing to the (private) frontend IP of the internal load balancer.
 - Our recommended method for this is to create an Azure private DNS zone and set it up in the Management Node as described [below](#). However, this could also be configured using an external DNS provider.
- Test that calls are working.

Setting up a private DNS zone in Azure

To create and use an Azure private DNS zone:

- Set up a private DNS zone in Azure for your Teams Connector domain:

The screenshot shows the Azure portal interface for a private DNS zone named 'pexip'. The left sidebar contains navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Virtual network links, Properties, Locks, Monitoring, Alerts, Metrics, Automation, Tasks (preview), Export template, and Support + troubleshooting. The main area displays the 'Essentials' section with details for the resource group 'px-infinity-deployment-rg', subscription 'Azure', and subscription ID 'd42ac30b-2c3a-42ae-9c36'. Below this is a table of record sets:

Name	Type	TTL	Value	Auto registered
@	SOA	3600	Email: azureprivatedns-host.microsoft.com Host: azureprivatedns.net Refresh: 3600 Retry: 300 Expire: 2419200 Minimum TTL: 10 Serial number: 1	False
conn.teams	A	3600	10.20.4.4	False

- Create an A record specifying your Teams Connector hostname (`$PxTeamsConnFqdn`) pointing to the (private) frontend IP address of the load balancer.
- Link your DNS zone to your Pexip Infinity deployment VNET (go to **Virtual Network Links** > **Add**).
- In the Pexip Infinity Management Node:
 - Add the Azure DNS zone as a DNS server (**System** > **DNS Servers** > **Add DNS Server**).
Use an IP address of **168.63.129.16** for the DNS virtual server (this is the Azure DNS IP address).

] pexip[Infinity Conferencing Platform

Status ▾History & Logs ▾System ▾Platform ▾Certificates ▾Call Control ▾Services ▾Users & Devices ▾One-Touch Join ▾Wel

Change System location

Name

Azure *

The name used to refer to this system location. Maximum length: 250 characters.

Description

A description of the system location. Maximum length: 250 characters.

DNS servers

Available DNS servers ⓘ

Q Filter

Choose all ⓘ

Chosen DNS servers ⓘ + *

168.63.129.16

Remove all ⓘ

The DNS servers to be used by Conferencing Nodes deployed in this Location. Hold down "Control", or "Command" on a Mac, to select more than one.

- b. Assign this DNS server to all the relevant system locations that need to resolve to your Teams Connector FQDN (the **Outgoing location** used by your Call Routing Rules used for Teams calls).

Setting up peering where the Pexip Infinity platform is in Azure

The example here shows how you would set up peering to the Teams Connector VNET via the Azure portal, if your Pexip Infinity platform is also in Azure:

Microsoft Azure

Home > Virtual networks > [redacted] Peerings >

Add peering

i For peering to work, two peering links must be created. By selecting remote virtual network, Azure will create both peering links.

This virtual network

Peering link name *

pexip-infinity-peering ✓

☒ Allow access to remote virtual network ⓘ

☐ Allow traffic to remote virtual network ⓘ

☐ Allow traffic forwarded from the remote virtual network (allow gateway transit) ⓘ

☐ Use remote virtual network gateway or route server ⓘ

Remote virtual network

Peering link name *

pexip-teams-connector-peering ✓

Virtual network deployment model ⓘ

☒ Resource manager

☐ Classic

☐ I know my resource ID ⓘ

Subscription * ⓘ

[redacted] ▼

Virtual network *

pexip-teams-connector-VNET ▼

☒ Allow access to current virtual network ⓘ

☐ Allow traffic to current virtual network ⓘ

☐ Allow traffic forwarded from current virtual network (allow gateway transit) ⓘ

☐ Use current virtual network gateway or route server ⓘ

Add

Configuring Pexip Infinity as a Microsoft Teams gateway

This section describes the configuration and administrative steps that are required within Pexip Infinity to route calls via the Teams Connector and operate as a Microsoft Teams gateway. It covers:

Prerequisites

Here are the prerequisites you must perform before configuring Pexip Infinity as a Microsoft Teams gateway.

Before configuring Pexip Infinity as a Microsoft Teams gateway, you should have:

- Performed a basic installation of the Pexip Infinity platform.
- Installed the Pexip Teams Connector as described in [Installing and configuring the Teams Connector in Azure](#).

Ensuring a Microsoft Teams license is installed

You must have a **teams** license enabled on your platform (**Platform > Licenses**).

It allows you to configure Microsoft Azure tenants and route calls to Microsoft Teams. The **teams** license controls how many Azure tenants can be configured; there is no limit to the number of Teams Connectors you can install or the number of instances within each Teams Connector.

If necessary, contact your Pexip authorized support representative to purchase the required license. Note that appropriate call licenses are also required for each gateway call that is placed into a Microsoft Teams conference.

Configuring global settings

You must ensure that the Pexip Infinity client API is enabled.

Go to **Platform > Global Settings** and in the **Connectivity** section ensure that **Enable support for Pexip Infinity Connect clients and Client API** is selected.

Configuring your Microsoft Teams tenants

A Microsoft Teams tenant is a dedicated instance of Azure Active Directory (Azure AD) that your organization obtains when it signs up for a Microsoft cloud service such as Azure or Office 365. You must configure the details of your Teams tenant — your Tenant ID specifically — in Pexip Infinity and then associate your Teams tenant with your Pexip Teams Connector so that Pexip Infinity can communicate with your Teams environment.

The number of tenants you can configure is controlled by your **teams** license. Service providers need to obtain an appropriate number of licenses for each tenant they are managing. Multiple Teams tenants (tenant IDs) can use the same Teams Connector (you must ensure that each Tenant ID has been consented to use the Pexip CVI app).

To configure your Teams tenant:

1. Go to **Call Control > Microsoft Teams Tenants** and select **Add Teams Tenant**.
2. Add the details of your tenant:

Name	The name that you specify here is used elsewhere in the Pexip Infinity interface when associating the tenant with a Teams Connector.
Description	An optional description of the Teams tenant.

Teams tenant ID	The Office 365 tenant ID where the Microsoft Teams users are homed.
	To retrieve your tenant ID from the Azure portal:
	a. Select Azure Active Directory .
	b. Under Manage , select Properties .
	c. The tenant ID is shown in the Directory ID field. You can use the Click to copy option to copy it.
	You can also check your tenant ID at https://www.whatismytenantid.com/ .

3. Select **Save**.

This Teams tenant is now available to associate with your Teams Connectors.

Configuring your Teams Connector (addresses, Event Hub, minimum capacity)

All communication between Pexip Infinity and Microsoft Teams is managed by your Pexip Teams Connector hosted in Azure.

To configure the address of your Teams Connector and decide whether to enable the Azure Event Hub (for scheduled scaling and enhanced status reporting):

1. Go to **Call Control > Microsoft Teams Connectors** and select **Add Microsoft Teams Connector**.
2. Add the details of your Teams Connector:

Name	The name that you specify here is used elsewhere in the Pexip Infinity interface when associating the Teams Connector with a system location, Call Routing Rule or Virtual Reception.
Description	An optional description of the Teams Connector.
Address of Teams Connector	The FQDN of the Teams Connector to use when placing outbound calls to Microsoft Teams. This is the address you stored in <code>\$PxTeamsConnFqdn</code> in the PowerShell variables initialization script, and is the FQDN (DNS name) of the Teams Connector load balancer in Azure that fronts the Teams Connector deployment e.g. <code>pexip-teamsconn-eu.teams.example.com</code> .
Port	The IP port to connect to on the Teams Connector.
Teams tenant	Select the Teams tenant to associate with this Teams Connector.
Enable Azure Event Hub	When enabled, this provides the ability to create scheduled scaling policies, and also provides enhanced status information for each Teams Connector node, such as call capacity and current media load.

Azure Event Hub connection string

When the Azure Event Hub is enabled you need to specify its connection string.

The connection string is in the format `Endpoint=sb://examplevmss-lltsgzoqun-ehn.servicebus.windows.net;/SharedAccessKeyName=pexip_teams_connector_access;SharedAccessKey=[string]`

To find this string in the Azure portal:

- Go to the static resource group for the Teams Connector (this has a name in the format `<prefix>-TeamsConn-<optional version>-<region>-static-RG`).
- Select the **Event Hubs Namespace** component (`<name>-EHN`).
- From the left-hand navigation menu, under **Settings** select **Shared access policies**.
- Select the `pexip_teams_connector_access` policy.
- Copy the **Connection string–primary key**.

SAS Policy: pexip_teams_conn... X

Save Discard Delete ...

☐ Manage

☒ Send

☒ Listen

Primary key

Secondary key

Connection string–primary key

Connection string–secondary key

You can also obtain it via the following PowerShell script (you must run your variables initialization script first):

```
$eventHub = (Get-AzResource -ResourceGroupName $PxTeamsConnStaticResourceGroupName -ResourceType
Microsoft.EventHub/namespaces)[0]
$eventHubKey = Get-AzEventHubKey -Name "pexip_teams_connector_access" -NamespaceName $eventHub.Name -ResourceGroupName
$eventHub.ResourceGroupName
Write-Host "The Azure Event Hub connection string' for $($PxTeamsConnFqdn) is:
$($eventHubKey.PrimaryConnectionString)"
```

Minimum number of instances

The minimum required number of instances in the Teams Connector scale set.

This setting only applies when the Azure Event Hub is enabled, and is the number of instances that are always running when there are no scaling policies in effect (it overrides the manual scaling settings in the Azure portal). See [Scheduling scaling and managing Teams Connector capacity](#) for more information.

If you increase this setting it can temporarily raise the "Scheduled scaling: some or all of the requested Teams Connector instances are not operational" alarm. It will last for a few minutes until the new instances are up and running. This is expected behavior and can occur with or without any scheduled scaling policies.

Default: 1

3. Select Save.

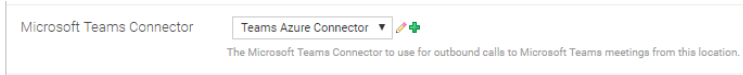
This Teams Connector is now available to associate with any system locations, Call Routing Rules or Virtual Receptionists that you configure (as described below) to handle Teams conferences.

Configuring a Teams Connector per location

You can optionally configure each system location with a Teams Connector to use by default for outbound calls to Teams meetings placed from Conferencing Nodes in that location. You can also explicitly configure individual Virtual Receptions or Call Routing Rules with a Teams Connector that will override the system location's Teams Connector.

To configure a location with a Teams Connector:

1. Go to **Platform > Locations** and select the required location.
2. In the **Microsoft Teams Connector** field, select the required Teams Connector from the drop-down list, and then select **Save**.



Configuring Virtual Receptions and Call Routing Rules for Teams meetings

You must configure the Call Routing Rules, and optionally the Virtual Receptions, that are required to route calls into Microsoft Teams meetings.

There are two ways you can configure Microsoft Teams gateway calls within Pexip Infinity:

- **Routing indirectly via a Virtual Reception:** here you configure Pexip Infinity to act as an IVR gateway or "lobby" by configuring a Virtual Reception to capture the Conference ID of the required conference, and then use a Call Routing Rule (typically the same rule as used for direct routing) to route the call into the Teams conference.
- **Routing directly via the Infinity Gateway:** here you only use one or more Call Routing Rules to route incoming calls for specific alias patterns — that will typically include the Conference ID — directly into the relevant Teams conferences.

i The Conference ID for a Teams meeting is between 9 and 12 digits long.

You will typically always configure the Virtual Reception indirect routing option, but depending on your requirements you may also choose to allow direct routing. The configuration required for these methods is explained below. Note that you always need to configure at least one Call Routing Rule whichever method(s) you use.

Routing indirectly via a Virtual Reception (IVR gateway)

To route calls to Teams conferences via a Virtual Reception (IVR gateway) you need:

- A Virtual Reception configured specifically to handle Microsoft Teams conferences.
- A Call Routing Rule to route the calls handled by the Virtual Reception into the relevant Teams conference. Typically you would configure the Virtual Reception and Call Routing Rule patterns so that the same rule can also support direct routing if required.

The Virtual Reception requests the caller to enter the Teams Conference ID which it then sends to the Teams Connector for verification. You can then optionally transform the Conference ID to meet any dial plan requirements, before the Infinity Gateway then matches the (optionally transformed) Conference ID and routes the caller to the appropriate Teams conference.

i The Virtual Reception details you enter here must correspond to the "Alternate VTC dialing instructions" that you specified in the **InstructionUri** switch in the **New-CsVideoInteropServiceProvider** command when installing the Teams Connector (see [Authorize trusted app to bypass Teams lobby and configure dialing instructions](#)).



Join the meeting directly

From a VTC/SIP system, enter: [23456789@example.com](tel:23456789@example.com)

Alternative addresses (e.g. H.323 systems): [23456789@198.51.100.40](tel:23456789@198.51.100.40) or [198.51.100.40##23456789](tel:198.51.100.40##23456789)

Join via lobby service

From a VTC/SIP system, enter: [teams@example.com](tel:teams@example.com) or [198.51.100.40](tel:198.51.100.40) then enter the conference ID: [23456789](tel:23456789) followed by #



From a browser

Go to: <https://px.vc.example.com/webapp/?conference=23456789@example.com>

Test call

To test your connection from a VTC/SIP system, enter: [testCall@example.com](tel:testCall@example.com) and verify that you can see and hear yourself

To configure the Virtual Reception:

1. Go to **Services > Virtual Receptions** and select **Add Virtual Reception**.
2. Configure the following fields (leave all other fields with default values or as required for your specific deployment):

Option	Description
Name	The name you will use to refer to this Virtual Reception, for example "Microsoft Teams IVR gateway".
Theme	If required, assign a customized theme to this Virtual Reception to brand it as the gateway to Teams conferences, for example by customizing the voice prompts or changing the appearance of the Virtual Reception splash screen.
Virtual Reception type	Select <i>Microsoft Teams</i> .
Teams Connector	Select the name of the Teams Connector to use to resolve Teams codes.
Lookup location	Specify the location that contains the Conferencing Nodes (typically Proxying Edge Nodes) that will communicate with the Teams Connector. These are the nodes referenced in the <code>\$PxNodeFqdns</code> variable in the initialization script.
Alias (#1)	Enter the alias that users will dial to use this Virtual Reception to place calls into Teams conferences. This must correspond to the <code>ivr</code> and <code>d</code> parameters that were specified in the <code>InstructionUri</code> switch in the <code>New-CsVideoInteropServiceProvider</code> command when installing the Teams Connector, i.e. the alias of the Virtual Reception is <code>ivr@d</code> , for example <code>teams@example.com</code> .

3. If you have specified the `ip` parameter in the `InstructionUri` switch in the `New-CsVideoInteropServiceProvider` command when installing the Teams Connector, then you must specify that IP address as an alias of this Virtual Reception.

Select **Add another Alias** and add **Alias (#2)**.

Alias (#2)	This must be the same IP address of the Conferencing Node that is specified in the <code>ip</code> parameter, e.g. <code>198.51.100.40</code> .
------------	---

4. Select **Save**.

To configure the associated Call Routing Rule:

- Configure the Call Routing Rule as described below for [direct and indirect routing](#).
- If you want to use a different rule for routing via a Virtual Reception than the rules you want to use for direct routing (e.g. because you want to limit the supported incoming call protocols), then follow the same principles as the direct routing rule, but use

different alias patterns in your Virtual Reception's **Post-lookup regex replace string** and your rule's **Destination alias regex match string**.

Call Routing Rules for direct and indirect routing

You need to create at least one Call Routing Rule, regardless of whether you are using indirect routing, direct routing, or both. The rule (s) must match the alias that has been captured — and probably transformed — by the Virtual Reception (for indirect routing) or match the alias originally dialed by the meeting participant (for direct routing).

Setting up direct routing

As with indirect routing, if you want to route calls to Teams conferences directly via the Infinity Gateway you need:

- To decide on an alias pattern that participants will dial to access the Teams conferences. The alias pattern will typically include the 9 to 12-digit Conference ID, for example the pattern could be just `<conference ID>` or `<conference ID>@<domain>`, for example `234567890@example.com` to access a Teams conference with a Conference ID of 234567890. See [Dial plan conflicts](#) if you have a more complicated dial plan to consider.
- A Call Routing Rule that matches that alias pattern and, if necessary, transforms it such that it contains just the Teams Conference ID which it can then use to connect to the conference.

Setting up the rule for direct and indirect routing

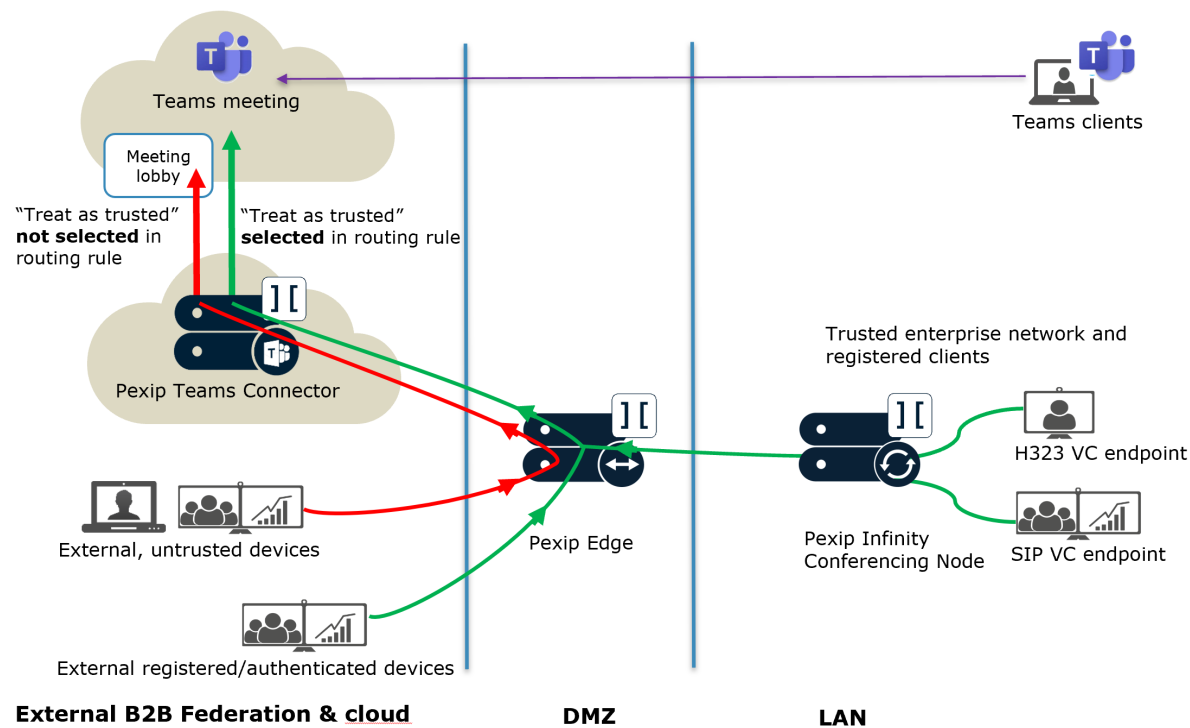
In the examples used here, the alias pattern that the rule needs to match is `<conference ID>` or `<conference ID>@<domain>` in both the indirect and direct routing scenarios. For direct routing this is the alias format that is dialed, and for indirect routing it is simply the Conference ID that is captured by the Virtual Reception.

The other consideration you must make when configuring your rule is whether the participant is trusted and therefore can be admitted directly into the Teams meeting (bypassing the Teams lobby) or whether the participant is untrusted and therefore has to be admitted into the meeting by an existing participant (held in the Teams lobby). You can decide to treat all participants as either trusted or untrusted if that is appropriate for your environment.

If you need to distinguish between trusted and untrusted participants, you must set up at least two Call Routing Rules where, typically, the first rule is configured to match participants who meet your trusted criteria, and the second rule could simply be those participants who did not match the first rule (and are therefore untrusted). The rule configuration options that you are most likely to use to determine whether it is a trusted participant or not are:

- **Calls being handled in location:** participants who are on an internal network for example, and whose calls are being handled by Conferencing Nodes in specific locations could be considered as trusted. Calls being received by Conferencing Nodes in locations that receive calls from an outside network or DMZ could be treated as external guest participants and thus not trusted.
- **Match incoming calls from registered devices only:** you could choose to trust any call being placed from a device that is registered to Pexip Infinity.

For rules configured to match against trusted participants, you should enable the rule's **Treat as trusted** option, which would cause Pexip Infinity to route the call directly into the Teams meeting (see [Authorize app to bypass Teams lobby and configure dialing instructions](#) for more information). The call is held in the meeting lobby if the rule does not have **Treat as trusted** enabled.



The other important rule configuration option to consider is:

- **Priority:** a number that determines the order in which the rules attempt to match the properties of the incoming call. Rules are processed in ascending number order, therefore, for example, you should ensure that any rules with **Match incoming calls from registered devices only** selected have a higher priority (lower number) than those rules where it is not selected. This is to avoid the call matching first against the "not selected" rule if all of the other rule settings are the same. Pexip Infinity's rule matching logic stops as soon as a matching rule is found, and this is why we recommend that your higher priority rules handle the "trusted" matching criteria first.

You can configure multiple "trusted" rules and multiple "untrusted" rules if necessary — if you have a complicated dial plan, for example, but we always recommended that you give your "trusted" rules a higher priority (lower number).

To configure your Call Routing Rules:

1. Go to **Services > Call Routing** and select **Add Call Routing Rule**.
2. The following table shows the fields to configure for your Call Routing Rule, and shows example values for three rules:
 - Rule #1: treats all devices on your enterprise network (handled by Conferencing Nodes in your "Internal network" location) as trusted.
 - Rule #2: treats any device that is registered to Pexip Infinity as trusted (regardless of the source location of the call).
 - Rule #3: treats any other call (not matching rule #1 or rule #2) as coming from an untrusted device.

(Leave all other fields with default values or as required for your specific deployment.)

Option	Description	Rule #1	Rule #2	Rule #3
Name	The name you will use to refer to this rule.	"Enterprise network"	"Registered devices"	"Unknown / guest"
Priority	Assign the priority for this rule.	20	40	60
Incoming gateway calls	Ensure this option is selected.	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Option	Description	Rule #1	Rule #2	Rule #3
Outgoing calls from a conference	Leave unselected.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Calls being handled in location	Applies the rule only if the incoming call is being handled by a Conferencing Node in the selected location. To apply the rule regardless of the location, select <i>Any Location</i> .	"Internal network" location	<i>Any location</i>	<i>Any location</i>
Match incoming calls from registered devices only	Select this option if you want this rule to only apply to calls placed from devices that are registered to Pexip Infinity. You will typically use this option in conjunction with the rule's <i>Treat as trusted</i> setting.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Match Infinity Connect Match SIP Match Lync / Skype for Business (MS-SIP) / Match H.323 / (Match Teams)	Select one or more of the match options as appropriate, depending on which types of systems/protocols you want to offer interoperability into Teams.	Select these options as appropriate		
Match against full alias URI	Leave unselected.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Destination alias regex match	Enter a regular expression that will match the calls to be sent to a Teams conference. For example, to match an alias in the form <conference ID> or <conference ID>@example.com, you could use: (\d{9,12})(@example\.com)? (Note that Teams Conference IDs are between 9 and 12 digits long. Putting ()? around the @domain part of the regex makes that part of the dial string optional.)	(\d{9,12})(@example\.com)?		
Destination alias regex replace string	If required, enter the regular expression string to transform the originally dialed (matched) alias into the 9 to 12-digit Conference ID to use to place the call into the Teams conference. If you do not need to change the alias, leave this field blank. When used with the example Destination alias regex match shown above you would use: \1 which would extract just the <conference ID> element of the alias.	\1		

Option	Description	Rule #1	Rule #2	Rule #3
Call capability	<p>To support incoming calls via SIP, H.323 and WebRTC/RTMP, <i>Same as incoming call</i> is recommended as this makes the most efficient use of Pexip Infinity resources.</p> <p>If you also want to support calls from Skype for Business / Lync, then you should select <i>Main video + presentation</i> instead to ensure that any SfB/Lync participants are able to escalate from audio-only to a video call after joining the conference (alternatively you can configure a separate rule just for matching incoming calls from Skype for Business / Lync and set that rule to use <i>Main video + presentation</i>).</p>	Select these options as appropriate		
Maximum call quality	<p>Controls the maximum call quality for participants connecting to this service:</p> <ul style="list-style-type: none"> ◦ <i>Use global setting</i>: use the global maximum call quality setting. ◦ <i>SD</i>: each participant is limited to SD quality. ◦ <i>HD</i>: each participant is limited to HD (720p) quality. ◦ <i>Full HD (1080p)</i>: allows any endpoint capable of Full HD to send and receive its main video at 1080p. <p>Default: <i>Use global setting</i></p>	Select these options as appropriate		
Theme	<p>If required, assign a customized theme to this rule (which will then be used for callers that use this rule to gateway into Teams). For example, the theme could use alternative labels on some of the splash screens that are displayed when connecting to a conference. You can also customize and enable the in-lobby and mute notifications, or the in-conference DTMF/keypad layout control options.</p>	Select these options as appropriate		
Call target	Select <i>Microsoft Teams meeting</i> .	<i>Microsoft Teams meeting</i>		
Outgoing location	Specify the location that contains the Conferencing Nodes (typically Proxying Edge Nodes) that will communicate with the Teams Connector. These are the nodes referenced in the <code>\$PxNodeFqdns</code> variable in the initialization script.	Select the location that contains the Conferencing Nodes that are referenced in the <code>\$PxNodeFqdns</code> variable in the initialization script		
Teams Connector	You can optionally select the Teams Connector you want to handle the call. If you do not specify anything, the Teams Connector associated with the outgoing location is used.	Leave unselected		
Treat as trusted	<p>Select <i>Treat as trusted</i> if you want Teams to treat the devices routed via this rule as part of the target organization for trust purposes.</p> <p>Typically you will select this option if the rule is handling devices that are trusted from Pexip Infinity's perspective, for example, you could treat the device as trusted if the caller is coming from a specific location, or if the device is registered to Pexip Infinity.</p> <p>Trusted participants are automatically admitted into a Teams conference.</p>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Option	Description	Rule #1	Rule #2	Rule #3
External participant avatar lookup	<p>This determines whether or not the Teams Connector requests from Exchange Online an avatar for each participant in the Teams conference.</p> <ul style="list-style-type: none"> ◦ Use global setting: use the global external participant avatar lookup setting. ◦ Yes: request the participant avatar from Exchange Online via the Teams Connector. ◦ No: use default avatar behavior. <p>Default: <i>Use global setting</i></p>		Select these options as appropriate	

3. Select Save.
4. Repeat the above steps adding your second and subsequent rules as appropriate.

Using the Microsoft Teams IVR gateway service

After the Virtual Reception and Call Routing Rule have been configured, third-party systems and devices can now dial the alias of the Virtual Reception (e.g. `teams@example.com`) and then, when prompted by the IVR service, enter the Conference ID of the Teams conference they want to join. The Infinity Gateway will then route the call into the appropriate Teams conference.

Using the direct gateway service

After the Call Routing Rule has been configured, third-party systems and devices can now dial an alias that matches your specified pattern (e.g. `234567890@example.com`) to be routed directly into the appropriate Teams conference (in this example the conference with a Conference ID of 234567890).

Dial plan conflicts

If the 9 to 12-digit Teams Conference ID and `<conference ID>@<domain>` style aliases might overlap with your existing dial plan, you can use something more specific for your aliases such as `teams.<conference ID>@<domain>` i.e. "teams." followed by the Conference ID and then the domain of your Pexip Infinity platform, for example `teams.234567890@example.com`.

In this case you would need to make the following adjustments to your Virtual Reception and Call Routing Rules (based on our examples above):

- In your Virtual Reception:
 - set **Post-lookup regex match** to `(.*)`
 - set **Post lookup regex replace string** to `teams.\1@example.com`
- In your Call Routing Rule(s):
 - set **Destination alias regex match** to `teams\.(\\d{9,12})@example\\.com`
 - leave **Destination alias regex replace string** set to `\1`

which will match everything entered into the Virtual Reception (which will be the Conference ID), and then prefix it with "teams." and append the "@example.com" domain, thus creating an alias in the form `teams.<conference ID>@<domain>`.

For example, if the caller enters 121212121 into the Virtual Reception the post-lookup match and replace strings shown here will convert that into `teams.121212121@example.com` which then needs to be matched by your Call Routing Rule.

which will match the alias pattern produced by the Virtual Reception and the new extended direct routing alias format.

With these changes, callers dialing the meeting directly would need to use this extended format of `teams.<conference ID>@<domain>`, but there is no change to callers going indirectly via the Virtual Reception (they would still dial `teams@example.com` and then enter the Conference ID as before).

You also need to configure a different set of "Alternate VTC dialing instructions" to use the `prefix` parameter to append "teams." to the displayed direct-dial addresses. In this case, our example `New-CsVideoInteropServiceProvider` command would look like this:

```
New-CsVideoInteropServiceProvider -Name Pexip -TenantKey "teams@example.com" -InstructionUri
"https://px.vc.example.com/teams/?conf={ConfId}&ivr=teams&d=example.com&ip=198.51.100.40&test=test_call&prefix=teams.&w&qrcode" -
AllowAppGuestJoinsAsAuthenticated $true - AadApplicationIds "c054d1cb-7961-48e1-b004-389e81356232"
```

If you have already configured the dialing instructions webpage, you can use the `Set-CsVideoInteropServiceProvider` command to amend it (see [Changing the alternative dialing instructions or IVR address](#)).

Enabling Microsoft Teams Rooms SIP/H.323 calling

You can optionally enable users who have Microsoft Teams Room devices with Pro licensing to make and receive 1:1 (also referred to as point-to-point) SIP/H.323 video calls with VTCs, and to call into Pexip Infinity VMRs.

There are several steps involved in enabling Microsoft Teams Rooms SIP/H.323 calling, as described in this section. It covers:

- [Licensing and software requirements](#)
- [Enabling your organization for Teams Room calling](#)
- [Enabling VTCs to call a Microsoft Teams Room](#)
- [Enabling a Microsoft Teams Room to call a VTC endpoint or VMR](#)

Licensing and software requirements

To use Teams Rooms SIP/H.323 calling you must adhere to the following requirements:

Pexip licensing requirements:

- Teams Room SIP/H.323 Calling plan license count to match your Microsoft Teams Room Pro license count (contact your Pexip authorized support representative for details)

Microsoft requirements:

- Teams Room Pro license
- Windows based device
- Teams Room software needs to be at least 4.17.51.0

Enabling your organization for Teams Room calling

This section describes how to enable your organization and Teams Room systems for SIP/H.323 calling.

Enabling and configuring your Teams calling policy

Your IT administrator must run some Teams PowerShell cmdlets to enable the Teams Room calling policy and give permissions to their VTCs that are to be used for 1:1 calling.

1. Start a PowerShell session as Administrator.
2. Run the following commands to sign in to your Teams tenant:
 - In all standard deployments run:


```
Import-Module MicrosoftTeams
Connect-MicrosoftTeams
```
 - If this is a GCC High / Azure US Government Cloud deployment then use these commands instead:


```
Import-Module MicrosoftTeams
Connect-MicrosoftTeams -TeamsEnvironmentName TeamsGCCH
```

You can then use the following commands to configure your calling policies.

To get a list of your existing policies, run the following command:

```
Get-CsTeamsRoomVideoTeleConferencingPolicy
```

To create a new policy:

```
New-CsTeamsRoomVideoTeleConferencingPolicy -Identity "MtrSipCallingPolicy" -Enabled $true -AreaCode "<CVI App ID>" -
ReceiveExternalCalls "Enabled" -ReceiveInternalCalls "Enabled"
```


where:

- **<CVI App ID>** is replaced with the CVI App ID of the Teams Connector to be contacted when the Teams Room places a call. (This means that you can have different policies assigned to different Teams Rooms within the same tenant. i.e. a tenant is not locked to a single logical Teams Connector App ID.)
- **"MtrSipCallingPolicy"** is the name of the policy being created. You can use any name of your choosing.

Note that `ReceiveExternalCalls` and `ReceiveInternalCalls` default to Disabled when not provided. These properties take only two values: Enabled or Disabled. This lets you configure which types of calls (marked as internal and/or externally originating) you can receive.

To grant a particular policy configuration to a user/room:

```
Grant-CsTeamsRoomVideoTeleConferencingPolicy -Identity "mtr01@pexample.com" -PolicyName "MtrSipCallingPolicy"
```

In the example above, "mtr01@pexample.com" is the Teams Room being configured within the "MtrSipCallingPolicy" policy.

Other useful commands

To update properties on an existing policy:

```
Set-CsTeamsRoomVideoTeleConferencingPolicy -Identity "MtrSipCallingPolicy" -ReceiveExternalCalls "Disabled"
```

The example above disables receiving external calls on the "MtrSipCallingPolicy" policy. Note that the `Identity` field is mandatory.

To view systems assigned with a policy:

```
Get-CsOnlineUser -Filter {TeamsRoomVideoTeleConferencingPolicy -eq "MtrSipCallingPolicy"} | Select-Object DisplayName, UserPrincipalName
```

The example above shows the systems assigned to the "MtrSipCallingPolicy" policy.

To delete a policy:

```
Remove-CsTeamsRoomVideoTeleConferencingPolicy -Identity "MtrSipCallingPolicy"
```

The example above deletes the "MtrSipCallingPolicy" policy.

Note that in `Set` and `Remove` commands the `-Identity` switch is the policy name, whereas in `Grant` commands the `-Identity` switch is the Teams Room being configured and the `-PolicyName` switch specifies the policy name.

Enabling VTCs to call a Microsoft Teams Room

To allow VTCs to call a Microsoft Teams Room you need to create a Call Routing Rule on Pexip Infinity with a **Call target of *Microsoft Teams Room***. In the same manner as the rules used to enable calls into Teams conferences, you can distinguish between trusted and untrusted (external) device types — this corresponds to the `ReceiveExternalCalls` and `ReceiveInternalCalls` settings in your `CsTeamsRoomVideoTeleConferencingPolicy` described above.

To configure your Call Routing Rules:

1. Go to **Services > Call Routing** and select **Add Call Routing Rule**.
2. The following table shows the fields to configure for your Call Routing Rule, and shows example values for three rules:
 - Rule #1: treats all devices on your enterprise network (handled by Conferencing Nodes in your "Internal network" location) as trusted.
 - Rule #2: treats any device that is registered to Pexip Infinity as trusted (regardless of the source location of the call).
 - Rule #3: treats any other call (not matching rule #1 or rule #2) as coming from an untrusted device.

(Leave all other fields with default values or as required for your specific deployment.)

Option	Description	Rule #1	Rule #2	Rule #3
Name	The name you will use to refer to this rule.	"To MTR from Enterprise network"	"To MTR from Registered devices"	"To MTR from Unknown / guest"
Priority	Assign the priority for this rule.	80	90	100

Option	Description	Rule #1	Rule #2	Rule #3
Incoming gateway calls	Ensure this option is selected.	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Outgoing calls from a conference	<p>If you only want to support 1:1 calls then leave this option unselected.</p> <p>However, if you want to allow a VMR to dial out to a Teams Room (including Automatically Dialed Participants), then select this option.</p>	Select these options as appropriate		
Calls being handled in location	<p>Applies the rule only if the incoming call is being handled by a Conferencing Node in the selected location.</p> <p>To apply the rule regardless of the location, select Any Location.</p>	"Internal network" location	<i>Any location</i>	<i>Any location</i>
Match incoming calls from registered devices only	Select this option if you want this rule to only apply to calls placed from devices that are registered to Pexip Infinity. You will typically use this option in conjunction with the rule's Treat as trusted setting.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Match Infinity Connect Match SIP Match Lync / Skype for Business (MS-SIP) / Match H.323 / (Match Teams)	Select one or more of the match options as appropriate, depending on which types of systems/protocols you want to offer interoperability into Teams.	Select these options as appropriate		
Match against full alias URI	Leave unselected.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Destination alias regex match	<p>Enter a regular expression that will match the calls addressed to a Microsoft Teams Room — this is the user principal (also referred to as the "calling URI") of a Teams room.</p> <p>For example, to match a UPN or alias in the form <code><username>@example.com</code>, you could use:</p> <p><code>.+@example\.com</code></p> <p>however, something more specific and appropriate for your dial plan would be better, and also consider the Priority of your rules.</p>	<code>.+@example\.com</code>		
Destination alias regex replace string	Leave blank			
Call target	Select Microsoft Teams Room .	Microsoft Teams Room		
Outgoing location	Specify the location that contains the Conferencing Nodes (typically Proxying Edge Nodes) that will communicate with the Teams Connector. These are the nodes referenced in the <code>\$PxNodeFqdns</code> variable in the initialization script.	Select the location that contains the Conferencing Nodes that are referenced in the <code>\$PxNodeFqdns</code> variable in the initialization script		
Teams Connector	You can optionally select the Teams Connector you want to handle the call. If you do not specify anything, the Teams Connector associated with the outgoing location is used.	Leave unselected		

Option	Description	Rule #1	Rule #2	Rule #3
Treat as trusted	<p>Select Treat as trusted if you want Teams to treat the devices routed via this rule as part of the target organization for trust purposes.</p> <p>Typically you will select this option if the rule is handling devices that are trusted from Pexip Infinity's perspective, for example, you could treat the device as trusted if the caller is coming from a specific location, or if the device is registered to Pexip Infinity.</p> <p>If the "trusted" flag is set, the call is classed as "internal", and if the "trusted" flag is not set, the call is classed as "external". This classification is used in conjunction with the Teams Room tenant policy to decide whether the call is delivered to a Teams Room.</p>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

3. Select Save.
4. Repeat the above steps adding your second and subsequent rules as appropriate.

Enabling a Microsoft Teams Room to call a VTC endpoint or VMR

There are some additional steps that must be followed to enable a Microsoft Teams Room to call a VTC or VMR via a Teams Connector, as described below.

Allow incoming calls on any firewalls protecting your Conferencing Nodes

To enable calls from Teams Rooms to reach Pexip Infinity, port 4277 must be enabled on any protecting firewalls for your Conferencing Nodes to allow inbound calls towards those nodes from your Teams Connector.

Enabling the Teams Connector to receive calls from a Teams Room

You must configure the Teams Connector Azure Bot to receive calls from a Teams Room.

To do this you can run the following PowerShell commands:


1. Run the variable initialization script. If you performing these steps in a new session you must also assign the \$AppId variable.
2. Run the following commands to configure the Azure Bot:

```
# Connect to Azure
Connect-AzAccount
# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1
# Enable incoming calls from a Teams Room
Enable-PexTeamsBotIncomingCallRoute -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxBotResourceGroupName -AppId $AppId -
TeamsConnectorFqdn $PxTeamsConnFqdn
```

The Teams Connector can now handle incoming calls from a Teams Room (and route them onwards to Pexip Infinity).

Calling into a VMR

No additional configuration is required to enable calls into a Pexip Infinity VMR — the Teams Room should just dial the VMR alias.

-  Note that Teams Rooms dialing into a Virtual Reception, Test Call Service, Media Playback Service or a VMR breakout room, or any other type of call that requires a call transfer, is not supported.

Calling a SIP or H.323 endpoint

To allow a Microsoft Teams Room to call a SIP or H.323 endpoint you need to create a Call Routing Rule on Pexip Infinity that matches incoming calls of type *Teams*.

To configure a Call Routing Rule:

1. Go to **Services > Call Routing** and select **Add Call Routing Rule**.
2. Configure the following fields (leave all other fields with default values or as required for your specific deployment):

Option	Description
Name	The name to use to refer to this rule e.g. "Incoming Teams Room calls".
Priority	Assign the priority for this rule.
Incoming gateway calls	Ensure this option is selected.
Outgoing calls from a conference	Leave unselected.
Calls being handled in location	Applies the rule only if the incoming call is being handled by a Conferencing Node in the selected location. To apply the rule regardless of the location, select <i>Any Location</i> .
Match Infinity Connect (WebRTC / RTMP) Match SIP Match Lync / Skype for Business (MS-SIP) Match H.323 Match Teams	Select <i>Match Teams</i> and deselect all of the other options.
Destination alias regex match	Enter a regular expression that will match the calls received from the Teams Room. For example, to match any alias in the vc.example.com domain: <code>.*@vc.example.com</code>
Destination alias regex replace string	If required, enter the regular expression string to transform the originally dialed (matched) alias into the alias to use to place the outbound call. If you do not need to change the alias, leave this field blank.
Call target	Select the appropriate target for the call, typically <i>Registered device or external system</i> or <i>Registered devices only</i> depending on your requirements.
Protocol SIP proxy H.323 gatekeeper	Select the protocol and associated proxy/gatekeeper as required if you are routing to external systems.

3. Select Save.

Enabling SIP Guest Join

Pexip's standard Microsoft Teams interoperability solution allows your own video conferencing endpoints (and your guests) to join Microsoft Teams meetings that you are hosting. The "SIP Guest Join" feature lets you join Microsoft Teams meetings with your own video conferencing endpoints where that meeting is being hosted by an external third-party organization who has not enabled Pexip interoperability themselves.

Note that this is a guest join experience, therefore someone has to be in the Microsoft Teams meeting to admit you from the lobby.

When using SIP Guest Join, your calls are routed via the Pexip Service, therefore you need to contact your Pexip authorized support representative to perform the necessary steps to enable the guest join process on your behalf. This includes:

- Purchasing a **Trusted Devices** license for the number of endpoints you own. Note that the Enterprise Room Connector (ERC) premium licensing package for Infinity includes SIP Guest Join (which allows for external Teams calls outside the organization, up to the purchased calling capacity).
- Providing the domains and IP addresses of **all** of the systems that will send SIP call signaling to the Pexip Service (typically these are the addresses of your externally-facing Conferencing Nodes, or VCS Expressway if relevant).

Note that you do not need to register your endpoints to the Pexip Service, or route any of your non-guest Teams interop calls via the Pexip Service.

Scope and limitations

This is a Teams commercial focused capability, thus all Teams meetings for Work or School i.e. used by enterprises and most governments in the commercial cloud are supported.

Note that the following are **not** supported:

- Teams meetings with "teams.live.com" (part of Teams for Home or Small Business)
- Teams meetings with "gov.teams.microsoft.us" (part of the Microsoft Government cloud)

Enabling guest join in Pexip Infinity

As a prerequisite to using the guest join feature you must be using One-Touch Join on the endpoints that you will use to join the Teams conferences.

To enable guest join you must create a suitable OTJ Meeting Processing Rule to handle invitations to Teams meetings, and ensure that your guest join calls are routed to the Pexip Service:

1. Add an OTJ Meeting Processing Rule (One-Touch Join > OTJ Meeting Processing Rules):

If you are using Pexip Infinity version 32 or later:

- You must add a new OTJ Meeting Processing Rule with a Meeting type of *Microsoft Teams SIP Guest Join*.

If you are using Pexip Infinity versions 29 to 31:

- You must add a new OTJ Meeting Processing Rule with a Meeting type of *Custom*.
- Use the following script in the rule's Custom template field:

```
{% set matches = pex_regex_search("https://teams.microsoft.com/(?:1/meetup-join|meet)/([A-z0-9-\\^\\%_.]+)/(\\[0-9+\\]\\)?context=([A-z0-9-\\^\\%_.]+)", calendar_event.body) %}
{% set context = "" %}
{% set organizerId = "" %}
{% set messageId = "" %}
{% set tenantId = "" %}
{% set threadID = "" %}

{% if matches %}
  {% set context = pex_regex_search("%7b%22id%22%3a%22([a-z0-9-]+)%22%2c%22oid%22%3a%22([a-z0-9-]+)%22%7d", matches[2]) %}
  {{{matches[0] ~ ", " ~ matches[1] ~ ", " ~ context[1]} | pex_base32 ~ "." ~ context[0]}}@pex.ms
{% else %}

  {% set matches = pex_regex_search("(https://teams.microsoft.com/meetingOptions.*?)"+"", calendar_event.body) %}
  {% set matches = matches[0] | replace("&", "&") %}

  {% set threadID = pex_regex_search("threadId(?:=|%3D)([a-zA-Z0-9_@.]+)", calendar_event.body) %}
  {% set threadID = threadID[0] | replace("_", ".", 1) %}

  {% set messageId = pex_regex_search("messageId(?:=|%3D)([a-zA-Z0-9_@.]+)", calendar_event.body) %}

  {% set organizerId = pex_regex_search("organizerId(?:=|%3D)([a-zA-Z0-9_@.]+)", calendar_event.body) %}


  {% set tenantId = pex_regex_search("tenantId(?:=|%3D)([a-zA-Z0-9_@.]+)", calendar_event.body) %}

  {% set threadID = pex_url_encode(('threadId', threadID)) %}

  {% set threadID = pex_regex_search("threadId=([a-zA-Z0-9_@.-%-]+)", threadID) %}

  {% if threadID[0] and messageId[0] and organizerId[0] and tenantId[0] %}
    {{{(threadID[0] ~ ", " ~ messageId[0] ~ ", " ~ organizerId[0]) | pex_base32 ~ "." ~ tenantId[0]}}@pex.ms
  {% endif %}
{% endif %}
```

(This rule looks for <https://teams.microsoft.com/<meetingcode>> and converts it into a SIP URI that can be routed via the Pexip Service.)

-  Consider the priorities of your rules. If you use Teams interop for your own Microsoft Teams meetings then your rules for those meetings should have a higher priority (lower number).
2. Ensure that the SIP Guest Join calls placed from your endpoint are routed to the Pexip Service.
- Typically you can do this via DNS as follows:

- a. Add a Call Routing Rule (**Services > Call Routing**).
- b. Configure the following fields (leave all other fields with default values or as required for your specific deployment):

Option	Description
Priority	Choose a suitable priority based on your existing rules.
Incoming gateway calls	Ensure this option is selected.
Outgoing calls from a conference	Leave unselected.
Match incoming calls from registered devices only	Select this option if you want the rule to apply only to calls received from devices that are registered to Pexip Infinity.
Match Infinity Connect (WebRTC / RTMP) Match SIP Match Lync / Skype for Business (MS-SIP) Match H.323 (Match Teams)	Select the relevant types of system/protocols you want to support, typically Match SIP or Match H.323 .
Match against full alias URI	Leave unselected
Destination alias regex match	Enter a regular expression that will match the Teams calls that are sent to the Pexip Service. We recommend: .*@pexip\.ms
Destination alias regex replace string	Leave blank.
Call target	Select <i>Registered device or external system</i> .
Protocol	Select <i>SIP</i> .
SIP proxy	Select <i>Use DNS</i> .

Controlling the layout seen by VTC participants

VTC participants see Pexip's standard 1+7 layout by default, but this can be changed to use other layouts. Meeting participants can also use DTMF/keypad controls to control the meeting layout during an ongoing conference, including switching to Microsoft's Large Gallery view, or participants using Cisco endpoints could also use the Pexip Meeting Controls macro.

Setting the default layout

Pexip's 1+7 layout is used by default for VTC participants within a Teams meeting, however this default can be changed to any of the other supported layouts.

The easiest way to change the default layout for all Teams gateway calls is to use local policy. This method also lets you control other layout options, such as the display of participant names, if required. You could, for example, use local policy to set the default layout to Adaptive Composition and then use one of the methods described below to dynamically change the layout to something else when and if required.

Note that the Teams Connector can only receive a maximum of 9 video streams from Teams, so, with the exception of the Adaptive Composition layout, you should not choose a layout that supports more than 9 participants.

This example local policy script enables the display of participant names and selects Adaptive Composition as the initial layout for all Microsoft Teams gateway calls:

```
{
  {% if service_config and service_config.service_type == "gateway" and service_config.called_device_type == "teams_conference" %}
    "action" : "continue",
    "result" : {{service_config|pex_update({"enable_overlay_text": true, "view":"five_mains_seven_pips"})|pex_to_json}}
  {% elif service_config %}
    "action" : "continue",
    "result" : {{service_config|pex_to_json}}
  {% else %}
    "action" : "reject",
    "result" : {}
  {% endif %}
}
```

If you want to apply the policy to a specific Call Routing Rule you can test against `service_config.name` (the Name of the rule), for example: `service_config.name == "name of routing rule"`

To use the 4 + 0 layout with Teams you would use: `"view":"four_mains_zero_pips"`.

Teams-like layout

To use the Teams-like layout you must use `"view":"teams"` and also enable the overlay text and active speaker indicators, as shown in the following script:

```
{
  {% if service_config and service_config.service_type == "gateway" and service_config.called_device_type == "teams_conference" %}
    "action" : "continue",
    "result" : {{service_config|pex_update({"enable_overlay_text": true, "view":"teams", "enable_active_speaker_indication":"true"})|pex_to_json}}
  {% elif service_config %}
    "action" : "continue",
    "result" : {{service_config|pex_to_json}}
  {% else %}
    "action" : "reject",
    "result" : {}
  {% endif %}
}
```

Note that:

- The layout cannot be customized via themes.
- It has the same hardware resource usage requirements as the Adaptive Composition layout.
- This layout is only suitable for use with Teams gateway calls.

Dynamically changing the layout during a conference

The most convenient way for a VTC participant to control the layout during a conference is to use DTMF/keypad commands directly from their endpoint. This allows the user to select which layout they receive, and to control how presentation content is displayed.

By default the layout changes every time the endpoint sends `*8` to the conference, and it cycles through all of the available Pexip layout types in this order: 1+7, Adaptive Composition, 1+21, 2+21, 2x2, 3x3, 4x4, 5x5, 1+0, 1+33. However, some of these layouts are unsuitable for use with Teams as they support more video streams than can be received by Pexip Infinity from Teams, and would result in "broken camera" images being displayed.

Microsoft's Large Gallery view

You can also introduce a DTMF conference control command — `"toggle_teams_large_gallery"`, which we suggest assigning to `*3` — into your themes that allows VTC participants to toggle between their current layout and Microsoft's Large Gallery view. Note that:

- There can sometimes be a short delay before the Large Gallery view takes effect.
- While in Large Gallery view, using `*8` to cycle layouts has no visible effect but you will see the effect if you toggle back out of Large Gallery view.

Using a customized theme

We recommend that you create a customized theme in Pexip Infinity that supports your required features, including:

- Limiting the DTMF/keypad controls to the ones that are suitable for use in a Teams conference.
- Selecting and specifying the sequence in which the layouts are displayed (by pressing *8).
- Adding the command to toggle between the current layout and Microsoft's Large Gallery view.

Here is an example theme configuration file (`themeconfig.json`) that is suitable for use with a Teams conference:

```
{
  "theme_version": 2,
  "dtmf_conference_control_commands": {
    "*3": "toggle_teams_large_gallery",
    "*4": "toggle_pres_in_mix",
    "*6": "toggle_self_mute",
    "*8": "cycle_layout",
    "*9": "toggle_multiscreen"
  },
  "dtmf_allowed_layouts": ["1:7", "ac", "2x2", "3x3", "1:0"]
}
```

This theme would override the default behavior and limit the DTMF controls to the following options that are relevant to a VTC connected to a Teams conference:

- ***3** command: toggles between the current layout and Microsoft's Large Gallery view
- ***4** command: toggles whether the presentation stream is sent to that endpoint in the main video mix (replacing some of the other video participants), or as a separate stream
- ***6** command: toggles the mute status of the participant
- ***8** command: cycles the layout through the set of available layouts as defined in `dtmf_allowed_layouts`
- ***9** command: toggles multiscreen participant display

It also limits the available layouts to Adaptive Composition ("ac") plus the other Pexip layouts that have a maximum of 9 or fewer video participants. If required, you can adapt the list to change the order, or to further reduce the available layouts by removing the layout types you do not want to use.

You can then apply this theme to your [Call Routing Rule\(s\)](#) used to route calls into Teams (or you can set it as your default theme if appropriate for your deployment).

Note that:

- You cannot use the Pexip branding portal to create your theme configuration file (`themeconfig.json`). You have to manually create your own theme configuration file (using the example above as the basis for this) and package it into a theme ZIP file.
- If multiple endpoints are gatewayed into the same Teams conference then they will all start with the same default layout, but any commands to change the layout received by the endpoint only apply to the endpoint issuing the commands.
- If you switch from a less resource-intensive layout (such as "1+7") to a more resource-intensive layout (such as "ac") this will consume additional Conferencing Node hardware resources.

See [Example themes](#) below for links to some example downloadable theme files.

Other layout control options

You can also use the Pexip APIs to control the layout:

- **Client API:** you can use the client API (typically via the Connect web app) to dynamically change the layout applied to a Teams gateway call. This involves using the `transform_layout` function with the client REST API or the `transformLayout` function with the PexRTC JavaScript client API.
- **Management API:** you can use the `transform_layout` management API command.
- **Cisco endpoint macros:** Cisco endpoints can use the Pexip Layout Controls macro that allows you to dynamically change the layout, or the Pexip Meeting Controls macro which offers an even greater range of conference control features in Teams meetings.

Using Adaptive Composition in a Teams conference

When Adaptive Composition is used with a Teams conference:

- Up to 9 video participants are shown in the Adaptive Composition layout seen by VTC participants.
- Each participant's video that is received from Teams for display to VTC participants is cropped and framed as appropriate.
- The VTC participant's video stream sent to Teams is cropped and framed as appropriate.
- Audio participant avatars are not supported.
- Inactive video participants are not removed from the video layout.
- Raised hand indicators are not supported.
- Spotlighting is not supported.

Customizing and enabling the in-lobby and mute notifications

You can customize the notifications that are displayed to gatewayed VTC participants when somebody is waiting in the Teams lobby, and you can also enable and customize the "Your mic has been muted" watermark icon that is displayed during a conference. You do this by customizing the theme that is associated with the Call Routing Rules used for [direct and indirect routing](#).

The following tables describe the relevant theme elements you can customize.



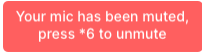
Settings in the `themeconfig.json` file:

Name	Content in Base theme	Usage guidance
<code>disable_conference_locked_indicator</code>	false	Set this to true if you want to disable the in-lobby notifications.
<code>conference_locked_indicator_text</code>	Conference locked	This is displayed when somebody enters the Teams lobby or there is a change in the count of waiting participants. We suggest you change it to something more appropriate, such as "Action required".
<code>conference_locked_indicator_n_waiting_text</code>	{number_of_waiting_participants} waiting for host	This is also displayed when somebody enters the Teams lobby or there is a change in the count of waiting participants. We suggest you change it to something more appropriate, such as "{number_of_waiting_participants} waiting in lobby".
<code>conference_unlocked_indicator_text</code>	Conference unlocked	This is displayed when the lobby is empty (everybody has been admitted, rejected, or left the lobby). We suggest you change it to something more appropriate, such as "The lobby is empty".
<code>disable_watermark_mute_icon</code>	true	Set this to false if you want to enable the "Your mic has been muted" notifications which are shown to a conference participant who is administratively muted.

For example, to implement the text label changes suggested above, and to enable muted indicators, you would configure your `themeconfig.json` file with:

```
{
  "theme_version": 2,
  "conference_locked_indicator_n_waiting_text": "{number_of_waiting_participants} waiting in lobby",
  "conference_locked_indicator_text": "Action required",
  "conference_unlocked_indicator_text": "The lobby is empty",
  "disable_watermark_mute_icon": false
}
```

In-conference graphic indicators:

File name	Content in Base theme	Usage guidance
icon_conference_locked.svg		Used in conjunction with the <code>conference_locked_indicator_text</code> and <code>conference_locked_indicator_n_waiting_text</code> labels when people are in the Teams lobby.
icon_conference_unlocked.svg		Used in conjunction with the <code>conference_unlocked_indicator_text</code> when the lobby becomes empty.
watermark_mute_icon.png		Used in conjunction with the <code>disable_watermark_mute_icon</code> setting, and is displayed when a conference participant is administratively muted.

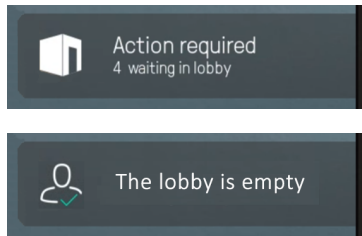
Audio files:

File name	Content in Base theme	Usage guidance
conf-participant_locked_out_48Khz_mono.wav	Three knocks	This is played to VTC participants when people are in the Teams lobby.

Example themes

We have provided some example themes that you can download from <https://dl.pexip.com/resources/themes/index.html>.

An example theme zip file, containing suggested themeconfig.json file settings for the in-lobby and mute notification, and some alternative svg graphics files for the locked/unlocked indicators is available in `lock-mute-indicator-theme.zip`. The indicators in the example theme look like this:



Another example theme, `lock-mute-indicator-dtmf-theme.zip`, extends the previous theme by including the recommended subset of [DTMF commands](#) that may be used to control a Teams conference.


DNS and ports requirements

You need to ensure that the endpoints and systems you want to gateway into Microsoft Teams can call into Pexip Infinity Conferencing Nodes, and that Conferencing Nodes can call out to Microsoft Teams.

Interoperability and deployment features

This section contains additional interoperability and deployment feature information.

- Each participant who is gatewayed via Pexip Infinity into a Teams conference consumes two call licenses (one for the inbound leg of the call and one for the outbound leg, as is standard for calls via the Infinity Gateway). Any external participants who are connected directly to the Teams conference do not consume a license.
- Bi-directional content sharing is supported between VTCs and Teams clients via VbSS:

- A Teams client can only share its screen or a window with a VTC system. However, VTC endpoints do see a splash screen if a Teams client starts PowerPoint Live sharing, or uses any other non-supported content share sources, such as Excel or Whiteboard:
 - If a VTC starts sharing content while it is held in the Teams lobby, when the VTC is admitted to the meeting the presentation will not be visible to the other participants in the meeting. The VTC participant would need to restart content sharing after being admitted into the meeting.
- A CVI participant's ability to share content in a Teams meeting depends on their role (Presenter or Attendee) in the meeting, and the "Who can present" setting in the meeting options:
 - A trusted CVI participant is always given a Presenter role in the Teams meeting (and thus can present). However, you can subsequently change that specific participant's role if required.
 - An untrusted CVI participant joins the meeting with the Presenter role only if the "Who can present" meeting setting is Everyone, otherwise it joins as an Attendee and cannot present.
 - If a CVI participant is an Attendee and attempts to share content from their endpoint, the sharing request is denied and the attendee is informed via a message on their endpoint such as "Sharing failed" (the exact message depends on the endpoint, and some endpoints may not display a message at all).
- Pexip provides the participant's display name if it is provided by the participant's device. This is independent of whether the device is trusted or not. If a display name is not provided, Pexip uses the device's SIP URI. Note that many devices use the system name as its display name. However, some older systems might choose, for example, to only provide the first part of the SIP URI (the part before the '@') as the display name.
- If a participant in a Teams meeting is spotlighted by a Teams client, that participant is also spotlighted in the layout presented to any gatewayed VTC system (except when using an Adaptive Composition layout).
- VTC endpoints that are gatewayed into a Teams meeting may sometimes be asked to send lower quality video (such as 180p). In these scenarios, even though Pexip Infinity / Teams Connector can support higher resolutions (and offers the higher resolutions to Teams), the video resolution requested by Teams can be lower — hence Pexip Infinity / Teams Connector sends the video format that Teams has requested. This is an operational behavior aspect of Teams and is beyond the control of Pexip software.
- The Teams Connector requests from Exchange Online an avatar for each participant in the conference. This avatar is used:
 - to represent any audio-only participant within the VTC's conference layout (replacing Pexip's standard audio indicator )
 - in the participant profile in the Connect app roster.

If the avatar request fails (the participant is not found in Exchange Online, or has no avatar) then a substitute graphic is generated based on the initials of the participant's name.
- Teams users joining with PSTN audio (callback and manual dial-in):
 - If a Teams user chooses to get a callback from the meeting to their PSTN device (instead of using computer audio) the user appears in the Pexip layout as a video participant and also as an audio-only participant with the avatar of that user. However, the video participant does not get voice-switched into main view when they speak.
 - If a Teams user selects "manual dial-in" they appear in the Pexip layout as an audio-only participant without that user's avatar as that audio participant has no association with the Teams user (and is also separate from any video connection).
- You cannot limit the **Maximum outbound call bandwidth** (the call leg towards the Teams Connector) — it is managed automatically by Pexip Infinity.
- If the Teams conference is recorded or transcribed, audio prompts indicating that recording/transcribing has been started/stopped are played, and indicators are included in the video stream sent to gatewayed participants.

Viewing Teams Connector instance, call and participant status


You can view Teams Connector instance, call and participant status via the Pexip Infinity Administrator interface:

- [Viewing the status of Teams Connector instances](#)
- [Monitoring calls placed into Teams conferences](#)

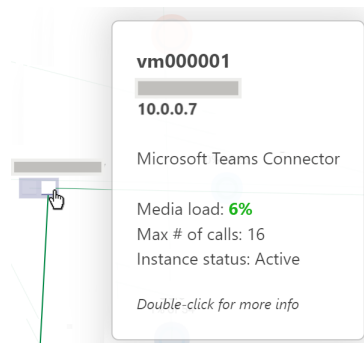
For general troubleshooting, call failures and installation issues, see [Troubleshooting Microsoft Teams and Pexip Infinity integrations](#).

Viewing the status of Teams Connector instances

You can view the status of each Teams Connector instance, such as call capacity and current media load, via the Pexip Infinity Administrator interface (**Status > Live View**).

The instances are represented by  icons. Each purple square represents a Teams Connector instance and the fill level of the square represents the current media load. A filled square in lighter purple represents an instance that is draining.

More details are shown when hovering over an instance, and you can double-click on an instance to see more detailed information about it.



The instance status could be:

- **Active:** the instance is operating normally.
- **Waiting for termination:** waiting for calls to clear so that the instance can be terminated.
- **Terminating:** there are no calls and the instance is terminating right now.
- **Maintenance:** the instance is not currently operational (e.g. it is restarting or being reconfigured).
- **Failure:** occurs when the instance had been in Maintenance status longer than expected. Typically this means that the instance is continually failing to restart.

You can also view a list of all deployed Teams Connector instances via **Status > Microsoft Teams Connectors**. This provides a summary of information including each node's name, IP address, maximum call capacity, current media load and the date/time of start-up and last update.

Note that the instance status information is only displayed if you have enabled the Azure Event Hub (**Call Control > Microsoft Teams Connectors** — see [Configuring your Teams Connector addresses](#) for more information). If the Azure Event Hub is not enabled then Live View only displays Teams Connector instances when a Teams call is in progress.

Monitoring calls placed into Teams conferences

When using the Pexip Infinity Administrator interface to monitor calls that are placed into Teams conferences, you should note that:

- Each participant who is gatewayed into a Teams conference is listed as a separate gateway call. However, if multiple participants are connected to the same Teams conference, the Live View (**Status > Live View**) shows them as connected to the same external conference — which is identified as "Teams meeting <conference ID>". The name/label shown for each call is based on the name of the associated Call Routing Rule.

- When viewing the status of the gateway call (**Status > Conferences**), the **Participants** tab also lists the other participants in the conference. The format of their alias indicates the type of participant:
 - <name>@<domain>** (email address) is used for any Teams Rooms or Teams clients in the call
 - trusted:<id>** represents another gatewayed participant who joined as a trusted participant
 - guest:<id>** represents another gatewayed participant who joined as an untrusted participant

These other participants do not have any associated media stream information.

Note that only the gatewayed participant is shown as consuming a port license. The outbound leg of the gateway call (into the Teams Connector), which consumes the second license of each gateway call, is not represented in the participant list.

The media streams associated with the call into the Teams meeting are shown against the conference backplane:

- There is one audio stream and multiple video streams. You also see a presentation stream if any participant is sharing content.
- Multiple video streams are set up to receive video from the Teams Connector to support the Pexip conference layout seen by gatewayed participants; if there are fewer participants than streams then the currently unused streams are shown as "Off stage".
- Pexip Infinity may simultaneously send up to 4 video streams and 4 presentation streams at different resolutions and frame rates to the Teams Connector, as requested by Teams.

Media streams

Type	Start time	Tx codec	Tx bitrate (kbps)	Tx resolution	Tx framerate	Tx packets sent	Tx packets lost	Tx current packet loss	Tx jitter (ms)	Rx codec	Rx bitrate (kbps)	Rx resolution	Rx framerate	Rx packets received	Rx packets lost	Rx current packet loss	Rx jitter (ms)
Video	2018-11-29 16:00:31 (GMT)	Off	0		0.0	0	0	0.0%	0.0	H.264 Baseline (mode 0)	728	640x360	25.0	248834	0	0.0%	0.2
Video	2018-11-29 16:00:31 (GMT)	Off	0		0.0	0	0	0.0%	0.0	H.264 Baseline (mode 0)	2216	1280x720	30.0	305836	0	0.0%	0.2
Video	2018-11-29 16:00:31 (GMT)	Off	0		0.0	0	0	0.0%	0.0	H.264 Baseline (mode 0)	792	640x360	25.0	24881	0	0.0%	0.4
Video	2018-11-29 16:00:31 (GMT)	H.264	787	640x360	30.0	686067	11	0.0%	1.3	Off	0		0.0	0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	H.264	2479	1280x720	30.0	31762	0	0.0%	2.2	Off	0		0.0	0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0.0	0	0	0.0%	0.0	Off stage	0		0.0	0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0.0	0	0	0.0%	0.0	Off stage	0		0.0	0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0.0	0	0	0.0%	0.0	Off stage	0		0.0	0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0.0	0	0	0.0%	0.0	Off stage	0		0.0	0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0.0	0	0	0.0%	0.0	Off stage	0		0.0	0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0.0	0	0	0.0%	0.0	Off stage	0		0.0	0	0	0.0%	0.0
Audio	2018-11-29 16:00:31 (GMT)	AAC-LD (64 kbit/s)	44			88313	2	0.0%	0.7	AAC-LD (64 kbit/s)	62			149768	0	0.0%	0.4

12 Media streams

- You cannot disconnect or transfer any of the other participants connected to the Teams conference.

Scheduling scaling and managing Teams Connector capacity

When you install a Teams Connector application in Azure, you initially specify the number of Teams Connector instances (VMs) to deploy, which determines the initial call capacity you can support. You can subsequently modify the number of instances to reflect changing capacity requirements. How you do this, and the amount of flexibility you have, depends upon whether you enable the Azure Event Hub in your Teams Connector configuration within Pexip Infinity.

Each Teams Connector instance can handle a maximum of 16 calls, although the capacity of each instance can vary depending on various factors such as call resolution, the number of presentation streams, and the number of participants in the Teams meetings. For capacity planning purposes we recommend that you assume 15 calls per instance.

If the Azure Event Hub **is** enabled, you can:

- Specify a minimum number of instances (capacity) that is always available.
- Use scheduled scaling via the Pexip Infinity Administrator interface to automatically scale up and down the capacity of your Teams Connector in Azure at different times of the day.

See [Using scheduled scaling](#) for details.

If the Azure Event Hub **is not** enabled, you can only:

- Manually configure the current number of instances via the Azure portal, when you want to adapt to changing capacity requirements.

See [Manual scaling via the Azure portal](#) for details.

Using scheduled scaling

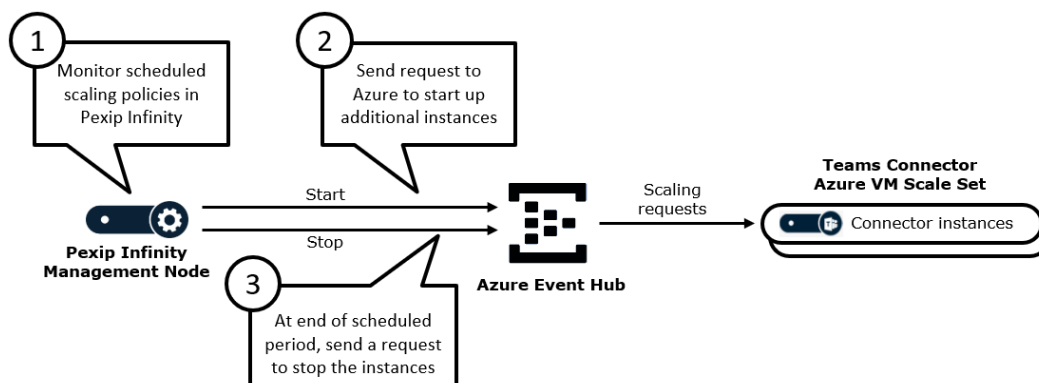
Scheduled scaling allows you to automatically scale up and down the capacity of your Teams Connector at different times of the day. This allows you, for example, to cater for increased demand during core working hours but just run a minimal capacity (and thus reduce running costs) at other times of the day.

To use scheduled scaling you must enable the Azure Event Hub in your Teams Connector configuration within Pexip Infinity (Call Control > Microsoft Teams Connectors).

How scheduled scaling works

To use scheduled scaling you need to define the [minimum number of instances](#), the [maximum number of instances](#), and [one or more scheduled scaling policies](#) within Pexip Infinity. These scaling policies define:

- The time and day(s) when the scaling policy comes into effect.
- How many additional Teams Connector instances to start up when the policy is in effect.



Pexip Infinity then monitors those policies:

1. Every minute, Pexip Infinity checks whether any scheduled scaling policies are to be applied at that time/day, taking into account the policy's **Minutes in advance** offset for the start time.
As it can take up to 20 minutes to start an instance, the offset is to allow enough time for all of the requested additional instances to be started up.
2. If a policy is to be applied, Pexip Infinity sends a request to Azure to start up the specified number of additional instances for the nominated Teams Connector.
 - The **Minimum number of instances** (configured against the Teams Connector in Pexip Infinity) defines the number of instances that are always running when there are no scaling policies in effect. The **Number of instances to add** (configured against each policy) is the amount of extra instances to start up.
 - If multiple policies are in force at the same time then Pexip Infinity will add all of the requested instances. For example, if the **Minimum number of instances** is 3 and there is a policy to add 2 more, and while that policy is in operation another policy to add 4 more comes into effect, then you will have 9 instances.
 - The maximum number of instances that can be running is capped by the instance count (slider) configuration in the Azure portal for the **Virtual machine scale set** in your Teams Connector resource group (see [Setting the maximum number of instances](#)). This limit is initialized by the `$PxTeamsConnInstanceCount` variable when you deploy a Teams Connector.
3. At the end of the scheduled time period, Pexip Infinity sends a request to Azure to turn off the additional instances, and Azure decides which excess instances to turn off.
 - The status of those instances is changed from **Active** to **Waiting for termination** and they will not accept any further calls.
 - Any existing calls will continue to be hosted on that instance, and the instance is only stopped when all calls have drained.
 - If a new policy comes into effect while some existing instances are waiting for termination then they may be restored to an active state, or brand new instances may be started up. This decision depends upon the current load of those instances and is to ensure that the expected capacity is available at the requested time.

Setting the minimum number of instances

To use scheduled scaling you need to define the minimum (baseline) number of instances. This is the number of instances that are always running when there are no scaling policies in effect. To do this:

1. Within the Pexip Infinity Administrator interface, go to **Call Control > Microsoft Teams Connectors** and select your Teams Connector.
2. Configure the **Minimum number of instances** as required.
3. **Save** your changes.

If you increase this setting it can temporarily raise the "Scheduled scaling: some or all of the requested Teams Connector instances are not operational" alarm. It will last for a few minutes until the new instances are up and running. This is expected behavior and can occur with or without any scheduled scaling policies.

Note that if the same Teams Connector is being used by multiple Teams tenants, then the minimum number of instances running in the scale set will be the highest of all of the individual **Minimum number of instances** settings for each tenant.

Setting the maximum number of instances

To change the maximum number of Teams Connector instances that can be running:

1. In the Azure portal, for your subscription, go to the **Virtual machine scale set** in your Teams Connector resource group.
2. In the **Settings** menu, select **Scaling**, and then select the **Configure** tab.
3. Ensure that **Manual scale** is selected (do not use autoscale).
4. Use the slider to increase/decrease the instance count as appropriate.

Note that if you reduce the instance count, any calls that are being handled by an instance that Azure chooses to delete will be dropped.

Home > connector-deployment-RG > nightlyconn

nightlyconn | Scaling ...

Virtual machine scale set

Search (Cmd+/)

Save Discard Refresh Logs Feedback

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Instances

Networking

Scaling

Disks

Operating system

Security

Guest + host updates

Size

Configure Scale-In Policy Run history JSON Notify Diagnostic settings

Autoscale is a built-in feature that helps applications perform their best when demand changes. You can choose to scale your resource manually to a specific instance count, or via a custom Autoscale policy that scales based on metric(s) thresholds, or schedule instance count which scales during designated time windows. Autoscale enables your resource to be performant and cost effective by adding and removing instances based on demand. [Learn more about Azure Autoscale](#) or [view the how-to video](#).

Choose how to scale your resource

Manual scale ☒ Maintain a fixed instance count

Custom autoscale ☐ Scale on any schedule, based on any metrics

Manual scale

Override condition


Instance count

Configuring scheduled scaling policies

You set up your scheduled scaling policies with Pexip Infinity. These are then monitored within Pexip Infinity and it sends instructions to Azure via the Azure Event Hub to start up or shut down the requested instances at the appropriate times as defined by the policy.

1. Within the Pexip Infinity Administrator interface, go to **Platform > Teams Connector Scheduled Scaling** and select **Add scheduled scaling policy**.

2. Configure the scheduled scaling policy:

Policy name	The name / description of this scheduled scaling policy.
Policy type	Select Teams Connector Scaling . (Currently this is the only policy type available.)
Teams Connector	Select the Teams Connector associated with this scheduling policy.
Number of instances to add	Enter the number of instances to add when this policy is in effect
Minutes in advance	The number of minutes in advance of the activation time to begin scaling up the instances.  This field currently defaults to 20 minutes and cannot be modified.
Enable this policy	Determines whether or not this scheduled scaling policy is enabled.
Timing	
Activation date	The date from which this policy applies. Enter a string in the format YYYY-MM-DD or use the associated calendar picker.
Time from	Specify the time of day when the additional instances are required. Enter a string in the format HH:MM:SS or choose one of the preconfigured options. Note that scaling up starts at the specified Minutes in advance of this time.
Time to	Specify the time of day when the additional instances should be stopped. Enter a string in the format HH:MM:SS or choose one of the preconfigured options.
Timezone	Select the timezone used to specify the activation date and time above.
Days	
Monday, Tuesday... etc.	Select the days of the week when the policy should be applied.

3. Select **Save**.

Manual scaling via the Azure portal

If you have **not** enabled the Azure Event Hub, you can modify the current number of Teams Connector instances at any time via the Azure portal, to reflect changing capacity requirements.

To change the current number of instances:

1. In the Azure portal, for your subscription, go to the **Virtual machine scale set** in your Teams Connector resource group.
 2. In the **Settings** menu, select **Scaling**, and then select the **Configure** tab.
 3. Ensure that **Manual scale** is selected (do not use autoscale).
 4. Use the slider to increase/decrease the instance count as appropriate.
- Note that if you reduce the instance count, any calls that are being handled by an instance that Azure chooses to delete will be dropped.

The screenshot shows the Azure portal interface for configuring the scaling of a virtual machine scale set named 'nightlyconn'. The breadcrumb navigation at the top indicates the path: Home > connector-deployment-RG > nightlyconn. The page title is 'nightlyconn | Scaling' with a subtitle 'Virtual machine scale set'. On the left, a sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Instances, Networking, Scaling (selected), Disks, Operating system, Security, Guest + host updates, and Size. The main content area has a top bar with 'Save', 'Discard', 'Refresh', 'Logs', and 'Feedback' buttons. Below this is a tabbed interface with 'Configure' (selected), 'Scale-In Policy', 'Run history', 'JSON', 'Notify', and 'Diagnostic settings'. A descriptive paragraph explains that Autoscale is a built-in feature for scaling based on demand, metrics, or schedules. The 'Choose how to scale your resource' section offers two options: 'Manual scale' (selected, with a radio button) and 'Custom autoscale'. The 'Manual scale' section includes an 'Override condition' field and a slider for 'Instance count' set to 5.

Note that:

- When the Azure Event Hub is enabled, this slider setting has a different purpose: it controls the **maximum** number of instances that can be running (instead of setting the current number).
- The **Minimum number of instances** (configured against the Teams Connector in Pexip Infinity) has no effect when the Azure Event Hub is not enabled.

Maintaining your Teams Connector deployment

After you have installed and configured your Teams Connector there are several maintenance tasks you may need to perform, such as adding extra Conferencing Nodes, changing your call capacity or upgrading the platform to the latest software version.

Modifying the Teams Connector's configuration


If you need to change the configuration on the Teams Connector after it has been deployed, you must make the necessary changes to the variables in your initialization script and then [redeploy](#) it using the new configuration, as described below. Configuration changes that require redeployment include:

- Changing the pool name or node FQDNs of the associated Conferencing Nodes (the name referenced by the `$PxNodeFqdns` variable in the initialization script).
- Enabling RDP access to the Teams Connector instances (from any addresses specified in the `$PxMgmtSrcAddrPrefixes` installation variable).
- Enabling support for dedicated hosting plans for Azure functions and VNet integration.

Note that you do not need to redeploy the Teams Connector if you need to change the Azure Network Security Group configuration, for example to add the IP address of a new Conferencing Node, update its [certificate](#), or if you need to modify the [number](#) of Teams Connector instances.

Upgrading the Teams Connector to the latest software

When upgrading the Teams Connector to version 33, you must [redeploy](#) it using your **original** variables initialization script (although you may apply any configuration changes, if required), and with the **latest** application software files and v33 redeployment script, as described below.

 When upgrading your Teams Connector to version 33:

New upgrade steps for version 33

- There are some new variables in the variables script to support Microsoft Teams Rooms SIP/H.323 calling:
 - `$PexipConfiguredConnectorFqdn` and `$PexipOutboundFqdn`.
 These new variables are passed as new parameters to `create_vmss_deployment.ps1` in the installation and redeploy scripts.
- There are some new variables in the variables script to support private routing:
 - `$PxExistingVNETResourceId` and `$PxUsePrivateRouting`.
 These new variables are passed as new parameters to `create_vmss_deployment.ps1` in the installation and redeploy scripts.
- Changes related to the support of certificate-based authentication of the CVI app:
 - A password-based authentication future deprecation notice has been added to the installation script.
 - The `create_vmss_deployment.ps1` command in the installation and redeploy scripts now takes `AppId` and `AppPassword` parameters instead of `AppCredential` (they still support password-based authentication).
- Other changes to the installation/redeployment scripts:
 - The commands to connect to MS Graph have been changed to use `PexTeamsMsGraph` in the `PexTeamsCviApplication` module.

Standard upgrade steps

- You must use the latest version of the [redeploy script](#) as contained here.
When using a new redeploy script, update the two lines in the redeploy script that say:


```
$AppId = ""
$AppPassword = ""
```

 to contain the Pexip CVI app ID and password from the version of the redeploy script you saved from your existing deployment.
- You must be using Az module version 9.0.1 minimum.
 - To check your installed version you can run:


```
Get-InstalledModule -Name Az -AllVersions
```

- To install the latest appropriate Az version you can run:

```
Install-Module -Name Az -MinimumVersion 9.0.1 -MaximumVersion 9.7.1 -AllowClobber -Scope AllUsers
```

- If you (the person performing the upgrade) did not perform the initial installation, you should ensure that you have all the relevant PowerShell modules and versions installed. If you are connecting to Azure Resource Manager / Microsoft Graph from your Windows PC for the first time, you must run the following PowerShell commands (as Administrator):


```
Install-Module -Name Az -MinimumVersion 9.0.1 -MaximumVersion 9.7.1 -AllowClobber -Scope AllUsers
Install-Module Microsoft.Graph -MinimumVersion 1.28.0 -MaximumVersion 2.2.0 -AllowClobber -Scope AllUsers
```

Note that:

- The installation of Microsoft Graph PowerShell SDK can take 5-10 minutes. Wait until you the get PS prompt back (several minutes after the install reports as completed) before continuing.
- The Az PowerShell module collects telemetry data (usage data) by default, however you can opt out from this data collection using `Disable-AzDataCollection` (see [this article](#) for more information).
- The Az PowerShell module remembers login information by default (i.e. you're not automatically logged out when closing your PowerShell window), but you can disable this with `Disable-AzContextAutosave` if required (see [this article](#) for more information).
- If you have deployed multiple Teams Connectors, you must follow the same redeploy process (with the appropriate variable initialization script) for each Teams Connector.
- As with all upgrades, you can continue to use the Pexip CVI app from your existing deployment.
- Your Pexip Infinity and Teams Connector installations must both be running the same software version (including minor/"dot" releases).

Using CBA with version 33

From version 33 you can use certificate-based authentication (CBA) to authenticate the Teams Connector CVI application towards MS Graph. In version 33, CBA is optional and the previous password-based authentication method is still the default mechanism.

-  The CBA method will be the default and recommended mechanism in version 34. Password-based authentication will still be supported in version 34 but we plan to deprecate it in a future release, thus we recommend migrating to CBA as soon as practicable.

If you want to start using CBA now, either when deploying Teams Connector for the first time, or when upgrading to version 33, see [Using certificate-based authentication for the Teams Connector CVI application](#).

Summary of script changes between versions

Version 32

Install/redeploy script:

- Microsoft Graph PowerShell is now used for creating the Teams Connector API app and for deploying the Teams Connector.
- The strong API app password generation is now performed by Azure AD.

Variables script:

- No changes.

Version 31

Install/redeploy script:

- The `create_vmss_deployment.ps1` command passes three new parameters: `$FunctionsDedicatedHostingPlan`, `$EventHubSourceAddressPrefixes` and `$VnetIntegration` (set to the new variables).
- A `Get-ChildItem -Recurse | Unblock-File` command has been added after `Set-ExecutionPolicy` to let you run PowerShell script files that were downloaded from the internet.

Variables script:

- There are three new variables: `$FunctionsDedicatedHostingPlan`, `$EventHubSourceAddressPrefixes` and `$VnetIntegration`. These are used to control support for dedicated hosting plans for Azure functions and VNet integration.
 - if you want to use the new features, set `$FunctionsDedicatedHostingPlan` and `$VnetIntegration` to `$true`, and specify the required IP addresses in the `$EventHubSourceAddressPrefixes` variable. The addresses specified in

`$EventHubSourceAddressPrefixes` can also be added or modified later in Azure if required.

- If you do not want to use the new features then you must add the following variable settings to your variables script:

```
$FunctionsDedicatedHostingPlan = $false
$EventHubSourceAddressPrefixes = @()
$VnetIntegration = $false
```

Version 30

Install/redeploy script:

- The Import-Module Az command now requires Az version 7.0.0 as a minimum.

Variables script:

- No changes.

Redeploying the Teams Connector

The Teams Connector redeployment process described below is required when you want to:

- Upgrade the Teams Connector software.
- Change the Teams Connector's configuration (such as the names of the associated Conferencing Nodes).
- Deploy a new Teams Connector in a different Azure region.

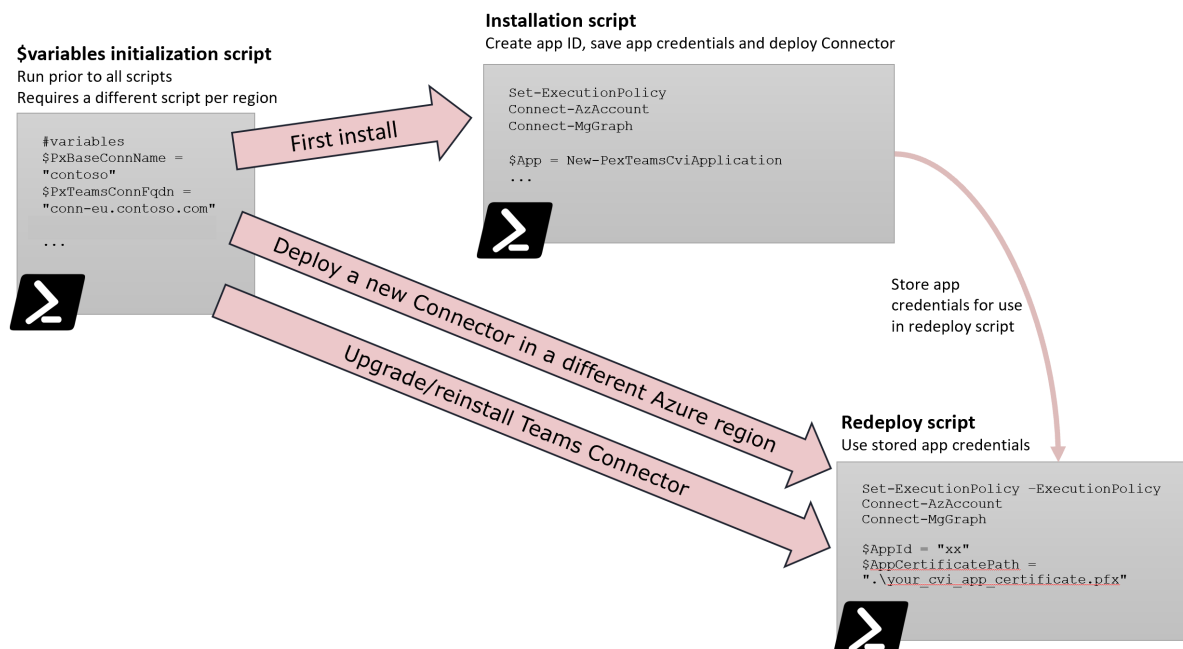
Note that there is no need to backup the Teams Connector as there are no specific settings, status or history information to preserve — if the deployment is lost you can just reinstall it with this redeploy process.

The redeployment process is summarized below:

1. Check that you have the latest relevant version of the Teams Connector software, and your variable initialization and redeploy scripts.
2. Delete the previous dynamic resource group and then recreate it for the new deployment, and ensure that the person performing the redeployment has the appropriate role for the resource groups in each region.
3. Reinstall the Teams Connector software.

- i** Before running your scripts we recommend that you check the Azure status (<https://status.azure.com>) for your region. Any ongoing issues in your region with the Azure services used by the Teams Connector could cause the script to fail.

The redeployment steps are shown in the following diagram and are then described in detail below.



Check Teams Connector software, retrieve your original scripts, and check your Azure environment

1. Download the latest relevant version of the **Teams Connector ZIP** file (Pexip_Infinity_Connector_For_Microsoft_Teams_v33_<build>.zip) from the [Pexip download page](#).
Ensure that the Teams Connector version you download is the same version as your Pexip Infinity deployment (including minor/"dot" releases).
2. Extract the files to a folder on a local drive on your administrator PC.
3. Add your [PFX certificate](#) for the Teams Connector to this folder.
4. Retrieve your saved copies of the initialization and redeploy scripts. You should have created and stored your version of these scripts after you completed your initial installation of your first Teams Connector.
5. Check your saved copy of the initialization script that sets the environmental variables:
 - If you are upgrading your Teams Connector, you should compare the saved copy of your initialization script against the current version of this script (as listed [here](#)), and adjust your script if necessary, for example if new variables have been added.
 - If you are redeploying and need to change any of the previous configuration you should also adjust your initialization script as required.
 - If you have Teams Connectors in many regions, ensure that you have the correct versions of the initialization scripts that set the regional variables to the appropriate values.
6. Check your saved copy of the redeploy script:
 - If you are upgrading to v33, ensure that you use the latest version of the [redeploy script](#) instead of your previously saved copy, but that you have updated it with your Pexip CVI App ID and credentials from your previously saved copy, as described immediately below.
 - Compare your version of the redeploy script to the one that is shown [below](#), and verify that your redeploy script was updated with the CVI App ID and password for your deployment (to ensure that upgrades/redeployments can be done using the same App ID):

Confirm that the redeploy script contains your Pexip CVI App ID and credentials:

- When the installation script ran, it generated some output that listed the CVI App ID and credentials, similar to this:

```
### CVI App ID and credentials MUST be saved in the redeploy script ###

### IMPORTANT - DEPRECATION NOTICE:
### A certificate-based authentication process for the CVI app will be default for the Teams Connector in v34
### The password authentication method will be deprecated

$AppId = "c054d1cb-7961-48e1-b004-389xxxx32"
$AppPassword = "vAuo2ZWuh8Do0538dhuhfGRH58gdreHHwDhff7635fdHHE64gHTDYwd66rEW77uRE383=="
```

- These output lines should have been used to replace the two existing lines in the redeploy script that say:

```
$AppId = ""
$AppPassword = ""
```

This means that when you run the redeploy script, you will reuse the CVI app and password that you created the first time.

- You should use the redeploy script in all scenarios except for when deploying a Teams Connector for the very first time for your Azure subscription i.e. you should use the redeploy script when upgrading, redeploying, or deploying a new Teams Connector in another region.
- You only need one version of this script — you can use the same redeploy script in every region if you have multiple Teams Connectors.

Remove and then recreate the dynamic resource group and ensure appropriate roles are assigned to the resource groups

- i** These steps **must** be performed by the **Owner** of the Azure subscription used for the Teams Connector.

1. Run the following PowerShell command to connect to Azure:

- In all standard deployments run:

```
Connect-AzAccount
```

- If this is a GCC High / Azure US Government Cloud deployment then use this command instead:

```
Connect-AzAccount -EnvironmentName AzureUSGovernment
```

Then follow the prompts to sign in to Azure.

2. Run the variable initialization script that you used when your first installed the Teams Connector (to set the required subscription, resource group name and region variables).

3. Ensure that you are using the Azure subscription for the Teams Connector:
`Set-AzContext -SubscriptionId $PxSubscriptionId`
4. Run the following script. It first deletes the existing dynamic resource group and then recreates it for the new/redeployed Teams Connector.

```
# Remove the existing dynamic resource groups
$PxTeamsConnResourceGroup = Get-AzResourceGroup -Name $PxTeamsConnResourceGroupName -Location $PxAzureLocation -ErrorAction
SilentlyContinue
if ($PxTeamsConnResourceGroup) {
    if ($PxUsePrivateRouting) {
        # Disassociate NSG from the vmss subnet
        $nsg = Get-AzNetworkSecurityGroup -ResourceGroupName $PxTeamsConnResourceGroupName -ErrorAction SilentlyContinue
        if ($null -ne $nsg) {
            $vmssSubnet = $nsg.Subnets | Where-Object { $_.Id.EndsWith("vmss-subnet") }
            if ($null -eq $vmssSubnet) {
                Write-Warning "NSG is not attached to the VMSS subnet"
            } else {
                $subnetConfig = Get-AzVirtualNetworkSubnetConfig -ResourceId $vmssSubnet.Id
                $existingVnet = Get-AzVirtualNetwork -Name $vmssSubnet.Id.Split('/')[8] -ResourceGroupName $vmssSubnet.Id.Split('/')[4]
                Set-AzVirtualNetworkSubnetConfig -Name $vmssSubnet.Id.Split('/')[1] -AddressPrefix $subnetConfig.AddressPrefix -VirtualNetwork
$existingVnet -NetworkSecurityGroupId $null -ServiceEndpoint @("Microsoft.Storage", "Microsoft.KeyVault", "Microsoft.EventHub") | Out-Null
                Write-Warning "The next command will disassociate subnet $($vmssSubnet.Id.Split('/')[1]) from NSG $($nsg.Name)" -WarningAction
                Inquire
                $existingVnet | Set-AzVirtualNetwork | Out-Null
            }
        }
    }
    $resources = Get-AzResource -ResourceGroupName $PxTeamsConnResourceGroupName
    $resNames = $resources | ForEach-Object { return "`n" + $_.Name + " " + $_.ResourceType }
    Write-Host "In order to redeploy Pexip Teams Connector, resource group $PxTeamsConnResourceGroupName has to be deleted. Resources:
$resNames"
    Remove-AzResourceGroup -Name $PxTeamsConnResourceGroupName | Out-Null
}
$PxTeamsConnResourceGroup = Get-AzResourceGroup -Name $PxTeamsConnResourceGroupName -Location $PxAzureLocation -ErrorAction
SilentlyContinue
if ($null -ne $PxTeamsConnResourceGroup) {
    Write-Warning "$PxTeamsConnResourceGroupName still exists, however it has to be removed before redeploying!"
}

# Create Resource group for Teams Connector VMSS (per region)
# This is region specific; ensure that the initial variables are set with the
# correct values for $PxAzureLocation, $PxTeamsConnFqdn and $PxVmssRegion
New-AzResourceGroup -Name $PxTeamsConnResourceGroupName -Location $PxAzureLocation -Tag $tags
```

5. Ensure that the person performing the redeploy/upgrade has the Azure **Owner** role for the static and dynamic resource groups, and **Contributor** role for the Azure Bot resource group. If you have deployed in multiple regions you must do this for all of the resource groups in each region.

If the Owner of the Azure subscription will be performing the redeploy/upgrade then this step can be skipped, otherwise the subscription Owner must assign the required roles to the person performing the upgrade. You do this by running the following commands:

```
New-AzRoleAssignment -SignInName <email_name> -RoleDefinitionName "Owner" -ResourceGroupName
$PxTeamsConnStaticResourceGroupName
New-AzRoleAssignment -SignInName <email_name> -RoleDefinitionName "Owner" -ResourceGroupName $PxTeamsConnResourceGroupName
New-AzRoleAssignment -SignInName <email_name> -RoleDefinitionName "Contributor" -ResourceGroupName $PxBotResourceGroupName
```

where **<email_name>** is the email address / user principal name of the person who will perform the remaining upgrade/redeploy steps; for example where `alice@example.com` will perform the upgrade/redeploy, the `SignInName` would be `-SignInName alice@example.com` for the commands listed above.

Note:

- In the future, if another person were to upgrade or redeploy the Teams Connector, that person would also have to be granted the appropriate roles for these resource groups.
- You can also use the Azure portal to check/assign permissions by selecting the resource group and using the **Access control (IAM)** option.

Redeploy the Teams Connector

To redeploy the Teams Connector:

1. Ensure that you have **Owner** permissions for the API app (check App registrations in the Azure portal); see [Assigning Owner permissions for the API app](#) if you do not have permission.
2. In PowerShell, run your initialization script that sets the environmental variables.
If you have Teams Connectors in many regions, ensure that you run the version of the initialization script that sets the regional variables to the appropriate values.
3. In PowerShell run your redeploy script.
You only need one version of this script — you can use the same script in every region if you have multiple Teams Connectors.

This is the redeploy script. It is a variation on the installation script that only performs the necessary commands to redeploy the Teams Connector. As with the initial installation, we recommend running each group of commands step-by-step within PowerShell.

```
# Ensure the correct script and software combination is being used
try {$PxConnMajorVersion = (Get-Content .\version.json -ErrorAction Stop | Out-String | ConvertFrom-Json).major} catch {Write-Warning "Can't find version.json file. Make sure you run the installation script from the folder into which you extracted the files from the Teams Connector ZIP"}

if ($PxConnMajorVersion -ne "33"){Write-Warning "The Connector version (extracted ZIP files) and this deployment script version do not match. Connector version = $PxConnMajorVersion. Deployment script version = 33"}

# Connect to PowerShell Azure Resource Manager account
# Set execution policy for the current PowerShell process, when prompted type A (Yes to All)
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# The Unblock-File cmdlet lets you run PowerShell script files that were downloaded from the internet.
# By default, these files are blocked to protect the computer from untrusted files.
Get-ChildItem -Recurse | Unblock-File

# Connect to Azure with an authenticated account for use with Azure Resource Manager (in same window to reuse variables)
Connect-AzAccount

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph

# Before running the following commands, update the following 2 lines/variables with the CVI App ID and password
# that were output when the installation script was run
$AppId = ""
$AppPassword = ""

# Change context to the Pexip Subscription and set the app credentials
Set-AzContext -SubscriptionId $PxSubscriptionId

$AppSecurePassword = ConvertTo-SecureString -AsPlainText $AppPassword -Force

# Virtual Machine Scale Set (VMSS) creation
# Provide credentials to be used as local user/password for Pexip Teams Connector VMs
# Create a password (using the initialization script variables) for the Windows VM
$PxWinAdminSecurePassword = ConvertTo-SecureString -AsPlainText $PxWinAdminPassword -Force
$PxWinAdminCred = New-Object System.Management.Automation.PSCredential -ArgumentList $PxWinAdminUser,$PxWinAdminSecurePassword

# Optionally if you did not want to specify the password as a variable, use Get-Credential
# $PxWinAdminCred = Get-Credential

# Deploy the Teams Connector VMs
# this step can take up to 30 minutes to complete
.\create_vmss_deployment.ps1 -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxTeamsConnResourceGroupName -VmssName
"$($PxBaseConnName)$($PxVmssRegion)" -VMAdminCredential $PxWinAdminCred -PfxPath $PxPfxCertFileName -TeamsConnectorFqdn $PxTeamsConnFqdn -
PexipFqdns $PxNodeFqdns -instanceCount $PxTeamsConnInstanceCount -AppId $AppId -AppPassword $AppSecurePassword -
StaticResourcesResourceGroupName $PxTeamsConnStaticResourceGroupName -IncidentReporting $PxTeamsConnIncidentReporting -RdpSourceAddressPrefixes
$PxMgmtSrcAddrPrefixes -PexipSourceAddressPrefixes $PxNodesSourceAddressPrefixes -WupdScheduledInstallDay $PxWupdScheduledInstallDay -
WupdScheduledInstallTime $PxWupdScheduledInstallTime -WupdActiveHoursStart $PxWupdActiveHoursStart -WupdActiveHoursEnd $PxWupdActiveHoursEnd -
CustomerUsageAttribution $PxCustomerUsageAttribution -UseAzureHybridBenefit $PxUseAzureHybridBenefit -Tag $tags -TeamsConnectorApiApplicationId
$TeamsConnectorApiApplicationId -FunctionsDedicatedHostingPlan $FunctionsDedicatedHostingPlan -EventHubSourceAddressPrefixes
$EventHubSourceAddressPrefixes -VnetIntegration $VnetIntegration -PexipConfiguredConnectorFqdn $PexipConfiguredConnectorFqdn -PexipOutboundFqdn
$PexipOutboundFqdn -ExistingVNETResourceId $PxExistingVNETResourceId -UsePrivateRouting $PxUsePrivateRouting

# supply the PFX certificate file password when prompted

# Please enter the password for the PFX certificate '.\xxxxxxxx.pfx': *****
```



```
# Printing finished message
Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host " All steps completed."
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host
```

```
# This script only applies to GCC High / Azure US Government Cloud deployments

# Ensure the correct script and software combination is being used
try {$PxConnMajorVersion = (Get-Content .\version.json -ErrorAction Stop | Out-String | ConvertFrom-Json).major} catch {Write-Warning "Can't find version.json file. Make sure you run the installation script from the folder into which you extracted the files from the Teams Connector ZIP"}

if ($PxConnMajorVersion -ne "33"){Write-Warning "The Connector version (extracted ZIP files) and this deployment script version do not match. Connector version = $PxConnMajorVersion. Deployment script version = 33"}

# Set VmImage variable to hold the CIS STIG image properties - STIG image is optional but typical
# In a later step in this script you can choose not to use the STIG image
$VmImage = @{
    "sku"           = "cis-win-2019-stig"
    "offer"         = "cis-win-2019-stig"
    "publisher"     = "center-for-internet-security-inc"
    "version"       = "latest"}

# Connect to PowerShell Azure Resource Manager account
# Set execution policy for the current PowerShell process, when prompted type A (Yes to All)
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# The Unblock-File cmdlet lets you run PowerShell script files that were downloaded from the internet.
# By default, these files are blocked to protect the computer from untrusted files.
Get-ChildItem -Recurse | Unblock-File

# Connect to Azure USGovernment with an authenticated account for use with Azure Resource Manager (in same window to reuse variables)
Connect-AzAccount -EnvironmentName AzureUSGovernment

# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph

# Before running the following commands, update the following 2 lines/variables with the CVI App ID and password
# that were output when the installation script was run
$AppId = ""
$AppPassword = ""

# Change context to the Pexip Subscription and set the app credentials
Set-AzContext -SubscriptionId $PxSubscriptionId

$AppSecurePassword = ConvertTo-SecureString -AsPlainText $AppPassword -Force

# Virtual Machine Scale Set (VMSS) creation
# Provide credentials to be used as local user/password for Pexip Teams Connector VMs
# Create a password (using the initialization script variables) for the Windows VM
$PxWinAdminSecurePassword = ConvertTo-SecureString -AsPlainText $PxWinAdminPassword -Force
$PxWinAdminCred = New-Object System.Management.Automation.PSCredential -ArgumentList $PxWinAdminUser,$PxWinAdminSecurePassword

# Optionally if you did not want to specify the password as a variable, use Get-Credential
# $PxWinAdminCred = Get-Credential

# Deploy the Teams Connector VMs
# this step can take up to 30 minutes to complete
# if you are not using a STIG image then remove the following parameter from this command: -VmImage $VmImage
```

```
.\create_vmss_deployment.ps1 -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxTeamsConnResourceGroupName -VmssName
"$($PxBaseConnName)$($PxVmssRegion)" -VMAdminCredential $PxWinAdminCred -PfxPath $PxPfxCertFileName -TeamsConnectorFqdn $PxTeamsConnFqdn -
PexipFqdns $PxNodeFqdns -instanceCount $PxTeamsConnInstanceCount -AppId $AppId -AppPassword $AppSecurePassword -
StaticResourcesResourceGroupName $PxTeamsConnStaticResourceGroupName -IncidentReporting $PxTeamsConnIncidentReporting -RdpSourceAddressPrefixes
$PxMgmtSrcAddrPrefixes -PexipSourceAddressPrefixes $PxNodesSourceAddressPrefixes -WupdScheduledInstallDay $PxWupdScheduledInstallDay -
WupdScheduledInstallTime $PxWupdScheduledInstallTime -WupdActiveHoursStart $PxWupdActiveHoursStart -WupdActiveHoursEnd $PxWupdActiveHoursEnd -
CustomerUsageAttribution $PxCustomerUsageAttribution -UseAzureHybridBenefit $PxUseAzureHybridBenefit -Tag $tags -TeamsConnectorApiApplicationId
$TeamsConnectorApiApplicationId -FunctionsDedicatedHostingPlan $FunctionsDedicatedHostingPlan -EventHubSourceAddressPrefixes
$EventHubSourceAddressPrefixes -VnetIntegration $VnetIntegration -VmImage $VmImage -TeamsEnvironmentName TeamsGCCHigh -
PexipConfiguredConnectorFqdn $PexipConfiguredConnectorFqdn -PexipOutboundFqdn $PexipOutboundFqdn -ExistingVNETResourceId
$PxExistingVNETResourceId -UsePrivateRouting $PxUsePrivateRouting

# supply the PFX certificate file password when prompted

# Please enter the password for the PFX certificate '.\xxxxxxxx.pfx': *****

# Printing finished message
Write-Host
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host " All steps completed."
Write-Host
Write-Host "`n-----`n"
Write-Host
Write-Host
```

Updating the Teams Connector TLS certificate or the CVI app certificate

The Teams Connector TLS certificate and the CVI app certificate (if using certificate-based authentication (CBA) to authenticate the Teams Connector CVI application towards MS Graph) are stored in the Azure key vault and can be updated without having to redeploy the Teams Connector. The old (existing) certificates can be updated when they are expiring or when they have already expired. We recommend updating the certificates before they expire to avoid service disruptions.

If you need to roll back to an older (but still valid) version of a certificate, you must follow these steps to import it as an additional certificate. Simply disabling the current certificate will not automatically revert to the previous version.

Note:

- An alarm is raised on Pexip Infinity when the Teams Connector TLS certificate is due to expire within the next 30 days, and if it has expired. Currently there are no alarms raised if the CVI app certificate is due to expire.
- You can use the `test_pfx_certificate.ps1` script (which is in the Teams Connector ZIP file) to verify the currently installed Teams Connector TLS certificate (you cannot use it on the CVI app certificate). The command takes the format (run your variables initialization script first):

```
test_pfx_certificate.ps1 -TeamsConnectorFqdn $PxTeamsConnFqdn -PfxPath $PxPfxCertFileName -PfxPassword <certificate password>
```
- If you need to create a new CVI app certificate, follow steps 1-7 of the instructions at [Migrating \(upgrading\) an existing Teams Connector using password-authentication to CBA](#).

To update either the Teams Connector TLS certificate or the CVI app certificate:

1. In the Azure portal, for your subscription, select the Teams Connector static resource group (which typically has a name in the style <prefix>-TeamsConn-<region>-static-RG).
 2. Ensure that you are an owner of the resource group.
 3. Select the **Key vault** in that resource group.
 4. Add a **Key Vault Certificates Officer** role (a4417e6f-fecd-4de8-b567-7b0420556985) for your user principal (see [this article](#) for instructions).
 5. Open the **Certificates** pane (under **Settings**).
- (Note that the screenshot examples below are for the Teams Connector TLS certificate.)

- i** If you have VNet integration enabled and you see a warning "Firewall is turned on and your client IP address is not authorized to access this key vault" then you need to temporarily add your client IP address in the key vault's **Networking settings** (**Firewall** > **+ Add your client IP address**).

6. Select the relevant certificate:
 - To update the Teams Connector TLS certificate: select the certificate named **pexip-teams-connector-certificate**.
 - To update the CVI app certificate: select the certificate named **pexip-cvi-app-certificate**.

(Don't create/import a new certificate under a different name.)

7. Select **New Version**.

Version	Thumbprint
CURRENT VERSION	
ca088b02ed6b41be99206a5aa94a3ca5	3C96144718FA2A470CFA39F07A8850B7F02F697E
OLDER VERSIONS	

8. Select **Import** as the creation method.

9. Select your new pfx certificate and provide a password if the certificate is password protected.
10. Import the certificate.
After importing a certificate any alarm in Pexip Infinity related to an expiring or expired Teams Connector TLS certificate will be lowered, but the new certificate is **not applied** across the VMSS yet.
 - i** There is no need to delete the expired certificate.
11. You have to reboot the scale set to apply the new certificate version.
To do this, in the Azure portal, for your subscription, select the Teams Connector dynamic resource group (which typically has a name in the style <prefix>-TeamsConn-<version>-<region>-RG where <version> is optional), then the Virtual Machine Scale Set and select the **Restart** option from the top menu.
 - i** Perform the reboot during out-of-hours — rebooting the scale set drops any existing calls.
12. After a few minutes, when the instances have restarted, verify that you can make calls and that the desired state configuration (from within your dynamic resource group, select the **Automation Account**, and then **State configuration DSC**) is green for all the **running** VMSS instances. (Some instances might be deallocated if using scheduled scaling — these instances will be provisioned with the new certificate on startup.)

We recommend that you set certificate contact notifications (see <https://docs.microsoft.com/en-us/azure/key-vault/certificates/about-certificates#certificate-contacts>) as they can warn you of certificates that are due to expire.

Assigning Owner permissions for the API app

The person who will be performing the deployment must have **Owner** permissions for the API app.

When necessary, the script below can be used by the current Owner to assign the Owner permission to another user.

Note that you must have run the variables script to assign the \$TeamsConnectorApiApplicationId variable and you must replace **<email_name>** in the script with the email address / user principal name of the person who will perform the remaining installation steps.

1. Run the following PowerShell command to connect to Azure:

- In all standard deployments run:

```
Connect-AzAccount
```

- If this is a GCC High / Azure US Government Cloud deployment then use this command instead:

```
Connect-AzAccount -EnvironmentName AzureUSGovernment
```

Then follow the prompts to sign in to Azure.

2. Run the variables script to assign the \$TeamsConnectorApiApplicationId variable.
3. Copy the script below:
 - a. You must replace **<email_name>** in the script with the email address / user principal name of the person who will perform the remaining installation steps.

For external users (guests in a tenant) the `-UserId` parameter should be in the form "`<emailname_domain>#EXT#@<tenant_url>`".

For example, for external users, if the external user has an email name of `guestname@guestdomain.com` and is a guest in `maintenant.com`, then you would specify (note the `_` replacing the `@` between `guestname` and `guestdomain.com`):

```
-UserId "guestname_guestdomain.com#EXT#@maintenant.onmicrosoft.com"
```

- b. Run the script.

```
# Import the PexTeamsCviApplication PowerShell module
Import-Module .\PexTeamsCviApplication.psm1

# Connect to Graph
Connect-PexTeamsMsGraph

# Get the API application registration reference
$apiApp = Get-MgApplication -Filter "AppId eq '${TeamsConnectorApiApplicationId}'"

# Get the assignee
# Replace <email_name> with the email address / user principal name of the person to perform the remaining installation steps
$user = Get-MgUser -UserId <email_name>

# Assign owner role for the API application registration to the assignee
$newOwner = @{
    "odata.id" = "($graphUrl.AbsoluteUri)v1.0/directoryObjects/($user.Id)"
}

New-MgApplicationOwnerByRef -ApplicationId $apiApp.Id -BodyParameter $newOwner

# Get the service principal associated with the API application
$apiAppSp = Get-MgServicePrincipal -Filter "AppId eq '${TeamsConnectorApiApplicationId}'"

# Assign owner role for the API application service principal to the assignee
New-MgServicePrincipalOwnerByRef -ServicePrincipalId $apiAppSp.Id -BodyParameter $newOwner
```

Maintaining access to the Admin Consent web page

Reinstating the Admin Consent web page

If you are upgrading and you need to maintain access to the Admin Consent web page (typically only necessary if you are a service provider), then after completing the commands in the redeploy script you should also run the following additional PowerShell commands to reinstate the Admin Consent web page:

```
Import-Module .\PexTeamsCviApplication.psm1
```

```
$AdminConsentUrl = Publish-PexTeamsAdminConsentWebSite -SubscriptionId $PxSubscriptionId -ResourceGroupName $PxBotResourceGroupName
-PxBaseConnName $PxBaseConnName -AppId $AppId -SourceAddressPrefixes $PxConsentSourceAddressPrefixes -Confirm:$false -Tag $tags
```

and then

```
Write-Host "To give consent to the CVI app, go to: $AdminConsentUrl"
```

Note that if you run the above commands in a new window/session you will first need to run the PowerShell variables initialization script.

Changing the workstation / management network addresses that can access the consent page

To change the workstation / management network addresses that can provide consent for the Teams Connector CVI app (the addresses that were initially specified in `$PxConsentSourceAddressPrefixes`):

1. In the Azure portal, go to the Azure bot resource group — this is called `<prefix>-TeamsBot-RG`.
2. Select the **App Service** object.
3. Under **Settings**, select **Networking**.
4. Select **Configure Access Restrictions**.
5. Modify the addresses as required and save your changes.

Adding or removing Conferencing Nodes

You can change the pool of Conferencing Nodes that communicates with the Teams Connector without having to redeploy the Teams Connector itself, providing that the certificate installed on any new Conferencing Nodes contains an identity that was in the `$PxNodeFqdns` initialization script variable when the Teams Connector was originally deployed.

For example, if you have a Teams Connector deployed as shown in [Certificate and DNS examples for a Microsoft Teams integration](#) i.e.

- two existing Conferencing Nodes called `px01.vc.example.com` and `px02.vc.example.com`
- both nodes have the same certificate installed with subject name `px.vc.example.com`, and altNames `px.vc.example.com`, `px01.vc.example.com` and `px02.vc.example.com`
- the Teams Connector was deployed with the `$PxNodeFqdns` variable set to `px.vc.example.com`

and you want to add a new Conferencing Node `px03.vc.example.com`, you would have to:

1. Install a certificate on the new Conferencing Node that contains the common `px.vc.example.com` identity.
To remain consistent with our example certificate naming policy this would mean creating a new certificate with a subject name of `px.vc.example.com` and altNames of `px.vc.example.com`, `px01.vc.example.com`, `px02.vc.example.com` and `px03.vc.example.com`. You would then install this certificate on the new node and the two existing nodes.
Note that purely from a Teams Connector perspective, the node certificate only needs to contain the identity specified in the `$PxNodeFqdns` variable, but if the same Conferencing Nodes handle SIP and Skype for Business signaling then the example certificate naming policy used here is more appropriate.
2. You may need to update the Network Security Group associated with your Teams Connector in Azure.
The "MCU-signalling-endpoint" rule (priority 120) specifies the IP addresses of the Conferencing Nodes that can communicate with the Teams Connector. These addresses were based on the value of the `$PxNodesSourceAddressPrefixes` initialization script variable. If individual IP addresses were specified then you need to add the IP address of the new Conferencing Node to this NSG rule. If the variable specified an IP subnet and the new node's address is within that subnet, then no changes are required.
3. If you had to update the NSG rule, you should add the same new address to the `$PxNodesSourceAddressPrefixes` variable in your stored initialization script to keep it in sync with your actual deployment in case it is needed for a future redeployment or upgrade.

Removing Conferencing Nodes

If you need to remove a Conferencing Node from the pool for any reason, you do not need to make any immediate changes to your Teams Connector / Azure configuration. However, you should make any necessary changes to the `$PxNodesSourceAddressPrefixes` variable in your stored initialization script to keep it in sync with your actual deployment.

Changing the alternative dialing instructions or IVR address

You can use the `Set-CsVideoInteropServiceProvider` command to change the alternative dialing instructions or the IVR address (TenantKey).

Changing the alternative dialing instructions

You can update the webpage of "Alternate VTC dialing instructions" that the recipient of the meeting invite can look at, by adding or removing some of the address types that are displayed.

You do this by using the `Set-CsVideoInteropServiceProvider` command and supplying a new `InstructionUri` parameter, following the same rules as described when you [first created the service provider](#).

In our original example, the service provider was created with the command:

```
New-CsVideoInteropServiceProvider -Name Pexip -TenantKey "teams@example.com" -InstructionUri
"https://px.vc.example.com/teams/?conf={ConfId}&ivr=teams&d=example.com&ip=198.51.100.40&test=test_call&w&qrcode" -
AllowAppGuestJoinsAsAuthenticated $true - AadApplicationIds "c054d1cb-7961-48e1-b004-389e81356232"
```

To change the instructions to remove the "Test call" option, for example, the revised `InstructionUri` parameter would be
`"https://px.vc.example.com/teams/?conf={ConfId}&ivr=teams&d=example.com&ip=198.51.100.40&w&qrcode"`

and the command to apply the revised `InstructionUri` parameter would be:


```
Set-CsVideoInteropServiceProvider -Identity Pexip -InstructionUri "https://px.vc.example.com/teams/?conf=
{ConfId}&ivr=teams&d=example.com&ip=198.51.100.40&w&qrcode"
```


Changing the IVR address (TenantKey)

The IVR address (`TenantKey`) is the alias that you assign to the Pexip Virtual Reception that is to act as the IVR gateway into the Teams meetings. In our original example this was set to `teams@example.com`.

To change the address you can use the `Set-CsVideoInteropServiceProvider` command and supply a new `TenantKey` parameter, for example:

```
set-CsVideoInteropServiceProvider -Identity Pexip -TenantKey "newaddress@example.com"
```

 Note that when using `Set-CsVideoInteropServiceProvider`, the first parameter is called `-Identity`, not `-Name`.

 These changes may take up to 6 hours to come into effect.

Changing the call capacity of a Teams Connector


When you install a Teams Connector CVI application in Azure, you initially specify the number of Teams Connector instances (VMs) to deploy.

You can subsequently modify the number of instances via the Azure portal, to reflect changing capacity requirements, or if you have enabled the Azure Event Hub you can also automate/schedule changes in call capacity to suit your requirements.

See [Scheduling scaling and managing Teams Connector capacity](#) for full details.

Changing management workstation access

When you deploy a Teams Connector, a Network Security Group (NSG) is created within Azure. The NSG includes an "RDP" rule (priority 1000) that controls RDP access to the Teams Connector instances from any addresses specified in the `$PxMgmtSrcAddrPrefixes` installation variable.

 If no addresses were specified then a Deny rule is created instead. If you want to subsequently allow RDP access you must do a full redeploy. You cannot just convert the Deny rule into an Allow rule (as other associated Teams Connector settings also have to be modified).

You can change this rule if you want to specify a different set of management workstation addresses. To do this:

1. In the Azure portal, for your subscription, go to the **Network security group** in your Teams Connector resource group.
2. Select rule **1000 RDP**.
3. Change the **Source IP addresses/CIDR ranges** as appropriate for the management workstation / network subnet addresses you want to allow RDP access.
4. Select **Save**.
5. Update the `$PxMgmtSrcAddrPrefixes` installation variable in your stored initialization script to match your changes applied to the NSG to keep the script in sync with your actual deployment.

Mandating the use of Availability Zones

By default the Teams Connector uses Azure Availability Zones if they are available in the selected region. However if you want to mandate the use of Availability Zones you can add an extra parameter to the call to `create_vmss_deployment.ps1` in your installation/redeployment script.

The parameter is `-UseAvailabilityZones` and can be set to either "BestEffort" (the default) or "Mandatory".

If you use "Mandatory" and Availability Zones are not available in the selected region then the deployment will fail with an error: "Availability zones not available for the selected region or the VM type is not available in the availability zones". See [Azure regions with Availability Zones](#) for the list of supported regions.

Monitoring the Teams Connector

This section contains optional configuration in Azure for administrators who want to collect metrics and Windows event logs into Azure log analytics workspace and Azure Monitor Metrics (in preview at the time of writing).

1. From within the Azure portal, go to the Teams Connector's dynamic resource group and select **+ Create**.
2. Use the search bar to find and select **Log Analytics Workspace**.
3. Select **Create** to add the Log Analytics Workspace in your dynamic resource group.
 - a. Your subscription and resource group should be preselected.
 - b. Enter a **Name**, for example "pexip-teams-connector-log-analytics".
 - c. Select a **Region**.
 - d. Select **Review + Create**.

Create Log Analytics workspace ...

Basics

Tags

Review + Create

A Log Analytics workspace is the basic management unit of Azure Monitor Logs. There are specific considerations you should take when creating a new Log Analytics workspace. [Learn more](#)

With Azure Monitor Logs you can easily store, retain, and query data collected from your monitored resources in Azure and other environments for valuable insights. A Log Analytics workspace is the logical storage unit where your log data is collected and stored.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Azure Px R&D Docs

Resource group * ⓘ

pexample-RG

[Create new](#)

Instance details

Name * ⓘ

pexip-teams-connector-log-analytics

Region * ⓘ

UK South

Review + Create

< Previous

Next : Tags >

- e. Select **Create** (after the validation stage completes).

4. Wait for the workspace deployment to complete and select **Go to resource**.
5. Go to **Agents management** and select **Data Collection Rules**.
6. Select **+ Create** to create a Data Collection Rule.
 - a. On the **Basics** tab:
 - i. Enter a **Name**, for example "pexip-teams-connector-dcr", and select a **Region**.
 - ii. Leave the **Platform Type** as **Windows**.
 - b. On the **Resources** tab:
 - i. Select **+ Add resources**.
 - ii. Use the search bar to find, select and **Apply** the **Virtual machine scale set** for your Teams Connector.
 - c. On the **Collect and deliver** tab:
 - i. Select **+ Add data source**.
 - ii. Choose a **Data source type** of **Performance counters**.
 - iii. Select the required performance counters.
 - iv. As a **Destination**, select **Azure Monitor Logs** and choose your log analytics workspace. You can also send these metrics to Azure Monitor Metrics (in preview at the time of writing).
 - v. Select **Add data source**.
 - vi. Add a second data source: repeat the above steps for a data source of **Windows event logs**.
 - vii. Select **Review + Create**, and then select **Create**.

Create Data Collection Rule ...

Data collection rule management

Basics Resources **Collect and deliver** Review + create

Configure which data sources to collect, and where to send the data to.

[+ Add data source](#)

Data source	Destination(s)
Performance counters	Azure Monitor Logs
Windows event logs	Azure Monitor Logs

[Review + create](#) [< Previous](#) [Next : Review + create >](#)

7. This installs extension AzureMonitorWindowsAgent onto your VMSS instances. The agent will start sending metrics and logs to your log analytics workspace, although this can take some time.

Reviewing your log analytics workspace status and metrics

From within your log analytics workspace resource, you can go to **Monitoring > Insights** (or **Settings > Agents management**) to check whether the agents are connected to your workspace.

Agents State

Healthy: 2, Unhealthy: 0

Azure resource	Resource type	Computer ID	Last heartbeat	Trend	OS	Category	Version	Computer Environment
nighlygw5/1	virtualMachineScaleSets	vm000001	3 mins		Windows (Windows Server 2019 Datacenter)	Azure Monitor Agent	1.7.0.0	Azure
nighlygw5/0	virtualMachineScaleSets	vm000000	1 mins		Windows (Windows Server 2019 Datacenter)	Azure Monitor Agent	1.7.0.0	Azure

In the Log analytics workspace you can query the event logs or performance metrics in the **General > Logs** pane.

Results

TimeGenerated [UTC]	Source	EventLog	Computer	EventCategory	EventLevel	EventLevelName	UserName	ParameterXml
8/19/2022, 11:53:57.483 AM	Microsoft-Windows-PerfMon	Application	vm000001	1	3	Warning	S-1-5-20	<Param=NETFramework\Param=Pa
8/19/2022, 11:53:57.490 AM	Microsoft-Windows-PerfMon	Application	vm000001	1	3	Warning	S-1-5-20	<Param=BITTS\Param=Param=C\Win
8/19/2022, 11:53:57.493 AM	Microsoft-Windows-PerfMon	Application	vm000001	1	3	Warning	S-1-5-20	<Param=Las\Param=Param=C\Wind
8/19/2022, 11:53:57.507 AM	Microsoft-Windows-PerfMon	Application	vm000001	1	2	Warning	N/A	<Param=3400000\Param=
8/19/2022, 11:53:57.533 AM	Microsoft-Windows-PerfMon	Application	vm000001	1	3	Warning	S-1-5-20	<Param=VeniAppl\Param=Param=C
8/19/2022, 11:55:28.183 AM	Microsoft-Windows-PerfMon	Application	vm000000	1	3	Warning	S-1-5-20	<Param=NETFramework\Param=Pa
8/19/2022, 11:55:28.187 AM	Microsoft-Windows-PerfMon	Application	vm000000	1	3	Warning	S-1-5-20	<Param=BITTS\Param=Param=C\Win
8/19/2022, 11:55:28.220 AM	Microsoft-Windows-PerfMon	Application	vm000000	1	2	Warning	N/A	<Param=Las\Param=Param=C\Wind
8/19/2022, 11:55:28.237 AM	Microsoft-Windows-PerfMon	Application	vm000000	1	3	Warning	S-1-5-20	<Param=VeniAppl\Param=Param=C
8/19/2022, 11:58:05.797 AM	Microsoft-Windows-PerfMon	Application	vm000000	1	2	Warning	N/A	<Param=3400000\Param=
8/19/2022, 12:01:05.973 PM	Microsoft-Windows-PerfMon	Application	vm000000	1	2	Warning	N/A	<Param=3400000\Param=
8/19/2022, 11:56:56.817 AM	Service Control Manager	System	vm000000	1	2	Warning	N/A	<Param=Connected Devices Platform
8/19/2022, 11:56:56.817 AM	Service Control Manager	System	vm000000	1	2	Warning	N/A	<Param=Connected Devices Platform
8/19/2022, 11:55:02.370 AM	AzureDiagnostics	Application	vm000000	1	2	Warning	N/A	<Param=failed to save ITS Directory sta
8/19/2022, 11:55:02.370 AM	AzureDiagnostics	Application	vm000000	1	2	Warning	N/A	<Param=System.Runtime.InteropServices
8/19/2022, 11:55:06.627 AM	Microsoft-Windows-PerfMon	Application	vm000000	1	2	Warning	N/A	<Param=3400000\Param=
8/19/2022, 11:59:35.867 AM	Microsoft-Windows-PerfMon	Application	vm000001	1	2	Warning	N/A	<Param=3400000\Param=

In **Azure Monitor > Metrics** you can view the metrics/logs of the VMSS by selecting the correct scope pointing to the resource itself. In **Metric** you can either choose "guest" or "host" metrics.

Avg (Memory) Available Bytes for nighlygw5

Scope: nighlygw5, Metric: Virtual Machine Guest (Memory) Available Bytes, Aggregation: Avg

The above is only an example of monitoring configuration. If you want to persist the log analytics workspace, you could deploy it in the static resource group and only recreate the data collection rules after each Teams Connector redeployment. Monitoring can also be set up against an already existing log analytics workspace. You can also collect custom performance counters and custom event logs.

Enabling diagnostic logs

Most of the Azure resources support sending of diagnostic logs to a specified destination. To enable some of these to output logs to log analytics workspace, see the following settings.

Key Vault

[Home](#) > [nightlyconn-bdadaqfkq-KV](#)

The screenshot shows the 'Diagnostic settings' page for a Key Vault resource named 'nightlyconn-bdadaqfkq-KV'. The left-hand navigation pane is expanded to 'Diagnostic settings'. The main content area shows a table with columns 'Name', 'Storage account', and 'Event hub'. The table is currently empty, displaying 'No diagnostic settings defined'. Below the table is a link '+ Add diagnostic setting'. A note states: 'Click 'Add Diagnostic setting' above to configure the collection of the following data:' followed by a bulleted list: 'Audit Logs', 'Azure Policy Evaluation Details', and 'AllMetrics'.

nightlyconn-bdadaqfkq-KV | Diagnostic settings

Key vault

Search

Refresh Feedback

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Access policies

Events

Objects

Keys

Secrets

Certificates

Settings

Access configuration

Networking

Microsoft Defender for Cloud

Properties

Locks

Monitoring

Alerts

Metrics

Diagnostic settings

Diagnostic settings are used to configure streaming export of platform logs and metrics for a resource to one or more independent destinations. [Learn more about diagnostic settings](#)

Diagnostic settings

Name	Storage account	Event hub
No diagnostic settings defined		

[+ Add diagnostic setting](#)

Click 'Add Diagnostic setting' above to configure the collection of the following data:

- Audit Logs
- Azure Policy Evaluation Details
- AllMetrics

Diagnostic setting ...

 Save  Discard  Delete  Feedback

A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. [Learn more about the different log categories and contents of those logs](#)

Diagnostic setting name *

Logs

Category groups ○

☐ audit ☒ allLogs

Categories

- ☒ Audit Logs
- ☒ Azure Policy Evaluation Details

Metrics

☒ AllMetrics

Destination details

☒ Send to Log Analytics workspace

Subscription

Log Analytics workspace

☐ Archive to a storage account☐ Stream to an event hub☐ Send to partner solution

Event Hub



nightlyvmss-fkfk6222uo-EHN | Diagnostic settings

Event Hubs Namespace

Search



Refresh



Feedback

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Events

Settings

Shared access policies

Scale

Geo-Recovery

Networking

Encryption

Configuration

Properties

Locks

Entities

Event Hubs

Schema Registry

Monitoring

Alerts

Metrics

Diagnostic settings

Diagnostic settings are used to configure streaming export of platform logs and metric independent destinations. [Learn more about diagnostic settings](#)

Diagnostic settings

Name	Storage account	E
No diagnostic settings defined		

[+ Add diagnostic setting](#)

Click 'Add Diagnostic setting' above to configure the collection of the following data:

- Archive Logs
- Operational Logs
- Auto Scale Logs
- Kafka Coordinator Logs
- Kafka User Error Logs
- VNet/IP Filtering Connection Logs
- Customer Managed Key Logs
- Runtime Audit Logs
- Application Metrics Logs
- AllMetrics

Diagnostic setting ...

Save Discard Delete Feedback

A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. [Learn more about the different log categories and contents of those logs](#)

Diagnostic setting name *

Logs

Category groups

☒ allLogs ☐ audit

Categories

- ☒ Archive Logs
- ☒ Operational Logs
- ☒ Auto Scale Logs
- ☒ Kafka Coordinator Logs
- ☒ Kafka User Error Logs
- ☒ VNet/IP Filtering Connection Logs
- ☒ Customer Managed Key Logs
- ☒ Runtime Audit Logs
- ☒ Application Metrics Logs

Metrics

☒ AllMetrics

Destination details

☒ Send to Log Analytics workspace

Subscription

Log Analytics workspace

☐ Archive to a storage account

☐ Stream to an event hub

☐ Send to partner solution

VMSS

- Go to **VMSS resource** > **properties** and copy resource ID.
- Go to **Log analytics workspace resource** > **properties** and copy resource ID.
- Decide on your data retention (specified in days).
- Enable diagnostic settings using PowerShell. Make sure you connect to the correct tenant and subscription beforehand.

```
$resourceId = {previously retrieved resource id}
$workspaceId = {previously retrieved workspace id}
# Modify retention in days based on your requirements
$retentionDays = 30
Set-AzDiagnosticSetting -ResourceId $resourceId -Enabled $True -WorkspaceId $workspaceId -RetentionEnabled $True -RetentionInDays $retentionDays -EnableLog $true -EnableMetrics $true
```

Data retention for the whole log analytics workspace

See [this article](#) for information about configuring the default workspace retention policy.

Uninstalling the Teams Connector

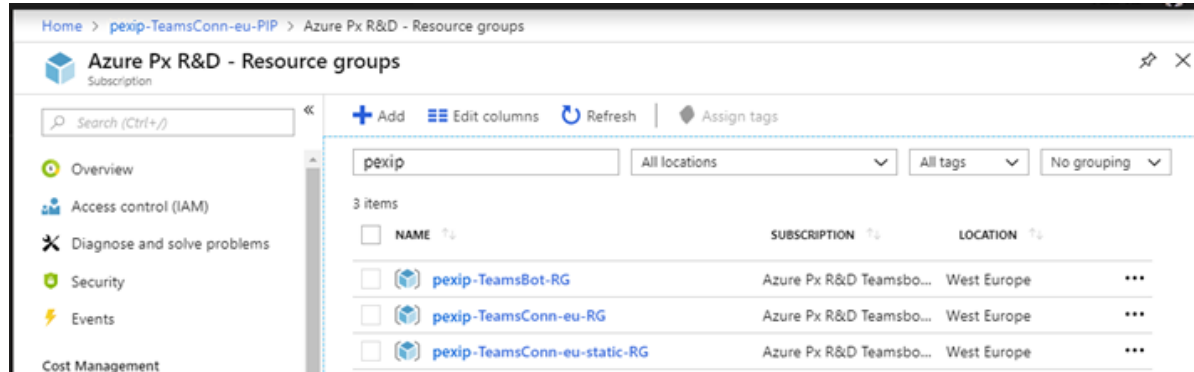
This section describes how to completely remove a Teams Connector installation, for example if you have a test system that you no longer need.

- Delete all of the resource groups from Azure:
 - Identify all of your existing Teams Connector resources in Azure: in the Azure portal, for your subscription, go to **Resource groups** and search for the prefix that you used when naming your resources — this is the value of the `$PxBaseConnName` variable in the initialization script.

You will see resource groups with names in the following formats:

- <prefix>-TeamsBot-RG: this contains the Azure Bot.
- <prefix>-TeamsConn-<region>-RG: this contains the Teams Connector instances (Virtual Machine scale set).
- <prefix>-TeamsConn-<region>-static-RG: this includes the Azure Key Vault and the public IP address of the Teams Connector.

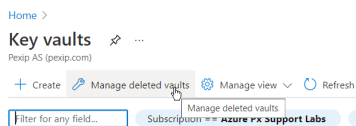
If you have deployed a Teams Connector in multiple regions you will see several <prefix>-TeamsConn-... resource groups.



- b. Select each resource group in turn (with the prefix of the Teams Connector you want to delete), and then select **Delete resource group** and confirm with the name of the resource group.

Note that as from v26, the Azure Key Vault is only soft-deleted when the static resource group is deleted. This means that if you subsequently redeploy your Teams Connector using the same variables initialization script, you will encounter deployment errors due to conflicts with key vault object names. To avoid this you must first purge the soft-deleted key vault:

- i. Go to the Key Vaults services within Azure.
- ii. Select **Manage deleted vaults**.
- iii. Select the subscription and purge the key vault.



2. Remove the App registrations from Azure:
 - a. Within the Azure portal, select **Azure Active Directory**.
 - b. Under **Manage**, select **App Registrations**.
 - c. Select the two app registrations (the CVI app and the API app) with the prefix of the Teams Connector you want to delete, and then **Delete** them.
3. Remove the Cloud Video Interop service provider from your tenant:
 - a. Start a PowerShell session and run the following commands to sign in to your Teams tenant:
 - In all standard deployments run:


```
Import-Module MicrosoftTeams
Connect-MicrosoftTeams
```
 - If this is a GCC High / Azure US Government Cloud deployment then use these commands instead:


```
Import-Module MicrosoftTeams
Connect-MicrosoftTeams -TeamsEnvironmentName TeamsGCCH
```
 - b. Run the command:


```
Remove-CsVideoInteropServiceProvider -Identity Pexip
```

To run the command, you need the **Global administrator** role.
4. Remove references to the Teams Connector from the Pexip Management Node:
 - a. Change or delete any locations, Virtual Reception or Call Routing Rules that are using the Teams Connector.
 - b. Go to **Call Control > Microsoft Teams Connectors** and delete the Teams Connector address.
5. Remove from DNS the record that refers to the FQDN of the Teams Connector load balancer.

Troubleshooting Microsoft Teams and Pexip Infinity integrations

This topic provides further information about the Teams Connector installation, describes any limitations and provides troubleshooting guidance when integrating Microsoft Teams with Pexip Infinity.

- [Installation issues](#)
- [Call failures \(invalid conference ID, rejected calls and disconnections\)](#)
- [Useful PowerShell commands](#)
- [Obtaining Teams Connector logs](#)
- [Data stored/processed by the Teams Connector](#)
- [Discovering your Azure tenant ID](#)
- [Viewing your tenant's app registrations](#)
- [Azure notification: Denial of service attack detected](#)

For information about Teams Connector status and troubleshooting failed calls, see [Viewing Teams Connector instance, call and participant status](#).

Installation issues

This table describes issues that might occur when running the Teams Connector installation script.

Issue / error message	Description / resolution
Invalid Bot data error message	If the PowerShell script output reports an error stating "InvalidBotData - Bot is not valid - Id is already in use", this means that the name of the Azure Bot you are attempting to create (via the Register-PexTeamsCviApplicationBot commands) is already in use elsewhere in Azure (most likely in another company's tenant).
Web space already exists error message	If the scripts fail to create a resource group with an error message in the style "Web space with name <name> already exists for subscription <id>", this typically means that a previous resource group of the same name had its contents deleted, but the resource group itself was not deleted. When removing resources from Azure e.g. prior to an upgrade or redeployment, ensure that the resource group itself is deleted.
Certificate error: CNG Key Storage Provider	If you see the error message "The private key of certificate '<thumbprint>' is stored in a CNG Key Storage Provider, which is currently not supported", this means that the RSA certificate has a private key in a CNG KSP (Cryptographic API: Next Generation Key Storage Provider), which is not supported. The private key must be stored in a Cryptographic Service Provider (CSP) of the CryptoAPI.
Certificate error: ECDSA certificate	If you see the error message "The certificate '<thumbprint>' is an ECDSA certificate, which is currently not supported", this means that you have an ECDSA certificate. The Teams Connector certificate must use RSA keys.
Get-AzADServicePrincipal: Insufficient privileges to complete the operation.	This error occurs if a guest user in the tenant attempts to deploy a Teams Connector, but the guest user does not have the Directory Readers role.

Issue / error message	Description / resolution
Invalid addresses	Ensure that any CIDR-style addresses are valid. For example, an "MCU-signalling-endpoint has invalid Address prefix" error would be reported if an invalid CIDR address, such as "91.90.42.96/26", was specified for the \$PxNodesSourceAddressPrefixes variable.
Other errors reported when running the create_vmss_deployment script	Many errors that are reported when you run create_vmss_deployment.ps1 are a result of typos or formatting errors in the variables initialization script, such as missing quotes around an IP address. Check the error messages produced by the script for more information about the nature of the problem.
More Teams Connector instances than expected are seen during the deployment process	Scale sets default to overprovisioning VMs. With overprovisioning turned on, the scale set spins up more VMs than requested, then deletes the extra VMs when the requested number of VMs are successfully provisioned. See https://docs.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-design-overview#overprovisioning for more information.
Connect-AzAccount WARNING: Unable to acquire token for tenant 'organizations' WARNING: Please run 'Connect-AzAccount -DeviceCode' if browser is not supported in this session.	<p>Try to Connect with: <code>Connect-AzAccount -DeviceCode</code></p> <p>If there are still issues then set default web proxy credentials and retry executing <code>Connect-AzAccount</code>:</p> <pre>[System.Net.WebRequest]::DefaultWebProxy.Credentials = [System.Net.CredentialCache]::DefaultCredentials</pre> <p>If that doesn't resolve the issue:</p> <ol style="list-style-type: none"> 1. Close all Powershell windows. 2. Open an Administrator command prompt. 3. Execute: <code>powershell -NoProfile -Command "Uninstall-Module Az.Accounts"</code> 4. Close the Administrator command prompt. 5. Open a PowerShell window. 6. Execute: <code>Install-Module Az.Accounts -MinimumVersion 1.9.5</code> 7. Retry <code>Connect-AzAccount</code>
The remote server returned an error: (407) Proxy Authentication Required	<p>Set default web proxy credentials and retry the deployment:</p> <pre>[System.Net.WebRequest]::DefaultWebProxy.Credentials = [System.Net.CredentialCache]::DefaultCredentials</pre>
Module being installed is not catalog signed	<p>This can happen when you run the <code>Install-Module</code> command if you have downgraded a package that was signed in the newer version.</p> <p>To overcome it you need to add the <code>-SkipPublisherCheck</code> switch to the <code>Install-Module</code> command.</p> <p>For more information see this article.</p>
WARNING: The names of some imported commands from the module 'Microsoft.Azure.PowerShell.Cmdlets.Network' include unapproved verbs that might make them less discoverable.	These warnings may appear and can be safely ignored.
and	
WARNING: The names of some imported commands from the module 'Az.Network' include unapproved verbs that might make them less discoverable.	

Issue / error message	Description / resolution
<p>Errors like:</p> <pre>Get-AzAccessToken : You must use multi-factor authentication to access resource MicrosoftGraphEndpointResourceId, please rerun 'Connect-AzAccount' with additional parameter '-AuthScope MicrosoftGraphEndpointResourceId'. At line:1 char:16 + ... cessToken = Get-AzAccessToken -ResourceTypeName MSGraph -TenantId \$ct ... + ~~~~~ + CategoryInfo : (:) [Get-AzAccessToken], AzPSAuthenticationFailedException + FullyQualifiedErrorId : Microsoft.Azure.Commands.Profile.GetAzureRmAccessTokenCommand</pre> <p>or a warning:</p> <pre>WARNING: Unable to acquire token for tenant {tenantId} with error 'You must use multi-factor authentication to access tenant {tenantId}, please rerun 'Connect-AzAccount' with additional parameter '-TenantId {tenantId}'.'</pre>	<p>You're probably connecting to a tenant as a guest user. Additionally the tenant has 2FA set up.</p> <p>The solution is to specify the <code>TenantId</code> parameter (where <code>TenantId</code> is an id of the tenant at which your user is a guest) when running <code>Connect-AzAccount</code>:</p> <pre>Connect-AzAccount -TenantId {tenantId}</pre>
<p>Errors like:</p> <pre>Get-MgApplication : Method not found: 'Void Microsoft.Graph.Authentication.AzureIdentityAccessTokenProvider..ctor (Azure.Core.TokenCredential, System.String[], Microsoft.Kiota.Authentication.Azure.ObservabilityOptions, System.String[])'. At line:1 char:1 + Get-MgApplication + ~~~~~ + CategoryInfo : (:) [Get-MgApplication_List], MissingMethodException + FullyQualifiedErrorId : Microsoft.Graph.PowerShell.Cmdlets.GetMgApplication_List</pre> <p>or</p> <pre>WARNING: Unable to acquire token for tenant 'organizations' with error 'Method not found: 'System.Threading.Tasks.Task`1<Azure.Identity.AuthenticationRecord> Azure.Identity.InteractiveBrowserCredential.AuthenticateAsync (Azure.Core.TokenRequestContext, System.Threading.CancellationToken)'. Connect-AzAccount : Method not found: 'System.Threading.Tasks.Task`1<Azure.Identity.AuthenticationRecord> Azure.Identity.InteractiveBrowserCredential.AuthenticateAsync (Azure.Core.TokenRequestContext, System.Threading.CancellationToken)'. At line:1 char:1 + Connect-AzAccount + ~~~~~ + CategoryInfo : (:) [Connect-AzAccount], MissingMethodException + FullyQualifiedErrorId : Microsoft.Azure.Commands.Profile.ConnectAzureRmAccountCommand</pre>	<p>These occur if you are using Microsoft.Graph module version 2.0.0. (See https://github.com/microsoftgraph/msgraph-sdk-powershell/issues/2148 for more information.)</p> <p>You should uninstall version 2.0.0 and install version 2.2.0 instead.</p>

Call failures (invalid conference ID, rejected calls and disconnections)

This section provides some guidance on how to troubleshoot failed calls to the Teams Connector.

As with all troubleshooting scenarios, reviewing the Pexip Infinity administrator log or the support log (where you can search for "support.teams" for specific Teams-related issues) may help you identify the possible cause of some failure scenarios.

Invalid conference ID failures when dialing via a Virtual Reception

The following table provides a set of typical causes of call failures and their associated solutions when the caller sees a "Conference ID invalid" or "Cannot connect to this extension" message when attempting to connect to a conference via a Virtual Reception.

Symptom	Possible cause	Resolution
Intermittent call failures: no unusual failures or error codes are in the logs.	The user is entering the wrong conference ID.	Ensure that the correct 9 to 12-digit conference ID is being entered.
Persistent call failures: no unusual failures or error codes are in the logs.	There is no Teams Connector configured against the Virtual Reception or the Virtual Reception's Lookup location, or the Conferencing Node cannot reach the nominated Teams Connector.	Ensure that a Teams Connector is configured against the Virtual Reception or the Lookup location, and that the Conferencing Nodes in that location can reach the nominated Teams Connector.
Support log entries report "Teams API request failed" and Error="401".	There is a problem with the certificate on either the Teams Connector or the Conferencing Node, for example it may have expired, been revoked, or the certificate subject names may not match against expected values.	Ensure that the Teams Connector and the Conferencing Nodes have a valid certificate that is signed by an external trusted CA. See Network and certificate requirements for information about certificate subject name requirements.
	The Conferencing Node is communicating with the wrong Teams Connector (if, for example, you have other Teams Connectors configured in different regions, or a lab system etc).	Ensure that the Virtual Reception is nominating the appropriate Outgoing location (and thus Conferencing Nodes) for the associated Teams Connector (i.e. the nodes that were referenced in the <code>\$PxNodeFqdns</code> variable in the initialization script for that Teams Connector).
	A chain of intermediate CA certificates installed on the Management Node (to provide the chain of trust for the Conferencing Node's certificate) includes a HTTP-to-HTTPS redirect in the AIA (Authority Information Access) portion of one of those intermediate certificates.	Obtain your certificates from a different Certificate Authority.
Support log entries report "Teams API request failed" and Error="500".	The wrong Azure tenant ID is configured in Pexip Infinity.	<p>Check the Azure tenant ID configured in Pexip Infinity (Call Control > Microsoft Teams Tenants) and which tenant ID is associated with the Teams Connector (Call Control > Microsoft Teams Connectors).</p> <p>This should be the tenant ID that was used during the process to authorize/consent the Pexip CVI app.</p>

Symptom	Possible cause	Resolution
CVI caller gets an invalid conference ID message when trying to join a scheduled meeting, but Teams clients can connect to the meeting.	<p>The meeting may have expired. This can happen if the meeting invitation was created a long time ago (typically more than 60 days ago). In this scenario, when the Teams user clicks on their meeting link, Teams automatically creates a new meeting with a new conference ID, and it appears as though they have joined the original meeting. The CVI caller cannot join as they do not have the new conference ID.</p> <p>If you have PSTN enabled on your meetings, you can see if a meeting has expired by joining the meeting via the Teams client and comparing the PSTN Conference ID shown in the Meeting Options join details against the PSTN Conference ID shown in the original invitation. If the two IDs are different then the original meeting has expired.</p> <p>See https://docs.microsoft.com/en-us/microsoftteams/limits-specifications-teams#meeting-expiration for more information.</p>	<p>The CVI caller cannot join that meeting.</p> <p>The only option is to create a new scheduled or instant meeting.</p>

Call is not connecting (direct or indirect dialing)

If the outbound leg of a gateway call to the Teams Connector fails, you typically see "Gateway dial out failed" or "Call rejected" in the participant history in Pexip Infinity. In these scenarios, Pexip Infinity has failed to place a call to the Teams Connector. This covers direct dialing (where the dialed alias includes the conference ID) and indirect dialing where the caller has successfully entered a valid conference ID into the Virtual Reception, but Pexip Infinity has then failed to connect the call.

The following table shows some possible causes and solutions for such failures:

Symptom	Possible cause	Resolution
Intermittent failures: disconnect reason is "Gateway dial out failed" or "Call rejected".	The dialed alias includes the wrong conference ID (direct dialing).	Ensure that the dialed alias contains the correct 9 to 12-digit conference ID.
	The Teams Connector was at maximum capacity and thus unable to take the call.	Consider increasing the number of instances in your Teams Connector (see Changing the call capacity of a Teams Connector).

Symptom	Possible cause	Resolution
Persistent failures: disconnect reason is "Gateway dial out failed" or "Call rejected".	The Call Routing Rule has an incorrect regex replace string.	Ensure that the Call Routing Rule regex replace string is extracting only the 9 to 12-digit conference ID.
	There is no Teams Connector configured against the Call Routing Rule or the rule's Outgoing location , or the Conferencing Node cannot reach the nominated Teams Connector.	Ensure that a Teams Connector is configured against the Call Routing Rule or the rule's Outgoing location , and that the Conferencing Nodes in that location can reach the nominated Teams Connector.
	The wrong tenant ID is configured in Pexip Infinity.	Check the Azure tenant ID configured in Pexip Infinity (Call Control > Microsoft Teams Tenants) and which tenant ID is associated with the Teams Connector (Call Control > Microsoft Teams Connectors). This should be the tenant ID that was used during the process to authorize/consent the Pexip CVI app.
Disconnect reason is "Gateway dial out failed" and the support log entries for the associated Call-id reports "Teams API request failed" and Error="401".	There is a problem with the certificate on either the Teams Connector or the Conferencing Node, for example it may have expired, been revoked, or the certificate subject names may not match against expected values.	Ensure that the Teams Connector and the Conferencing Nodes have a valid certificate that is signed by an external trusted CA. See Network and certificate requirements for information about certificate subject name requirements.
	The Conferencing Node is communicating with the wrong Teams Connector (if, for example, you have other Teams Connectors configured in different regions, or a lab system etc).	Ensure that the Call Routing Rules that are handling the calls are nominating the appropriate Outgoing location (and thus Conferencing Nodes) for the associated Teams Connector (i.e. the nodes that were referenced in the <code>\$PxNodeFqdns</code> variable in the initialization script for that Teams Connector).
	A chain of intermediate CA certificates installed on the Management Node (to provide the chain of trust for the Conferencing Node's certificate) includes a HTTP-to-HTTPS redirect in the AIA (Authority Information Access) portion of one of those intermediate certificates.	Obtain your certificates from a different Certificate Authority.

Disconnections from ongoing conferences

The following table shows some possible causes and solutions if participants are disconnected from ongoing conferences:

Symptom	Possible cause	Resolution
Calls are disconnected from a conference and "Running conference has been forcibly terminated due to MCU shutdown" messages are in the Pexip Infinity administrator log.	This can occur if the Adaptive Composition layout is in use and there is a large number of active participants.	This can be mitigated by assigning more RAM to your Conferencing Node processor cores.

Useful PowerShell commands

The following PowerShell commands are available when troubleshooting or maintaining your system:

Requirement	PowerShell command
List the existing interop service provider	<code>Get-CsVideoInteropServiceProvider</code>
Modify an interop service provider (see Changing the alternative dialing instructions or IVR address for more information)	<code>Set-CsVideoInteropServiceProvider</code>
Remove an interop provider	<code>Remove-CsVideoInteropServiceProvider</code>
List the existing service policies	<code>Get-CsTeamsVideoInteropServicePolicy -Filter "*enabled*"</code>
Return user information	<code>Get-CsOnlineUser -Identity username@example.com</code>
Enable interop for a named user	<code>Grant-CsTeamsVideoInteropServicePolicy -PolicyName PexipServiceProviderEnabled -Identity username@example.com</code>
Remove interop completely for a named user	<code>Grant-CsTeamsVideoInteropServicePolicy -PolicyName ServiceProviderDisabled -Identity username@example.com</code>
Remove any explicit interop policy for a named user i.e. revert to using the global default policy	<code>Grant-CsTeamsVideoInteropServicePolicy -PolicyName \$null -Identity username@example.com</code>
Revert the global interop policy to its default configuration	<code>Grant-CsTeamsVideoInteropServicePolicy -PolicyName ServiceProviderDisabled</code>
List all active CVI users	<code>Get-CsOnlineUser -Filter {TeamsVideoInteropServicePolicy -eq "PexipServiceProviderEnabled"} Select-Object DisplayName, UserPrincipalName, TeamsVideoInteropServicePolicy</code>

Obtaining Teams Connector logs

To help troubleshoot issues, your Pexip authorized support representative may ask you for log files from your Teams Connector in addition to a snapshot from your Pexip Infinity Management Node.

As of version 30, logs are stored in a **diag** container (previously **logs** container), and access is now via user delegated SAS tokens (previously SAS tokens signed by the account keys).

Note that logs and incident reports are kept for 30 days on the Teams Connector.

To get the logs and snapshot:

1. Get the Teams Connector logs:
 - a. In the Azure portal, for your subscription, select the Teams Connector static resource group that contains resources such as the key vault and public IP address.
 - b. Select the **Storage** account in that resource group.
 - c. Select **Access Control (IAM)** and assign the **Storage Blob Data Reader** role to the storage account for the user (you would need to do this even for yourself) who is going to generate the user delegated SAS token in order to access the **diag** container.
 - d. If you have VNet integration enabled you need to add both your client IP address and "176.125.235.150" in the storage account's **Networking** settings (**Firewall > Address range**).
 - e. Wait for 5 minutes for the access permission to propagate.
 - f. Generate a user delegated SAS token:
 - i. Go to **Data storage > Containers**.
 - ii. Click the context menu next to the **diag** container and select **Generate SAS**.

Name	Last modified	Public access level	Lease state	
<input type="checkbox"/> acrs	1/2/2020, 10:56:34 AM	Private	Available	...
<input type="checkbox"/> app	9/28/2018, 12:29:10 ...	Private	Available	...
<input checked="" type="checkbox"/> diag	7/16/2022, 12:38:12 ...	Private		...
<input type="checkbox"/> logs	3/2/2021, 9:45:44 AM	Private		...

Container properties
 Generate SAS
 Access policy
 Acquire lease
 Break lease
 Change access level
 Edit metadata
 Delete

- g. On the Generate SAS panel:
 - i. Select **User delegation key**.
 - ii. For **Permissions**, select **Read and List**.
 - iii. Set the **Allowed IP addresses** to 176.125.235.150 and HTTPS only.
 - iv. Click **Generate SAS token and URL**.
 - v. Copy the **Blob SAS URL** that appears. You will use this in Pexportal.

Generate SAS



A shared access signature (SAS) is a URI that grants restricted access to an Azure Storage container. Use it when you want to grant access to storage account resources for a specific time range without sharing your storage account key. [Learn more about creating an account SAS](#)

Signing method

☐ Account key ☒ User delegation key

Permissions * ⓘ

2 selected ▼

Start and expiry date/time ⓘ

Start

08/03/2022

3:04:26 PM

(UTC+00:00) Dublin, Edinburgh, Lisbon, London ▼

Expiry

08/03/2022

11:04:26 PM

(UTC+00:00) Dublin, Edinburgh, Lisbon, London ▼

Allowed IP addresses ⓘ

176.125.235.150 ✓

Allowed protocols ⓘ

☒ HTTPS only ☐ HTTPS and HTTP

Generate SAS token and URL

Blob SAS token ⓘ

sp=rl&st=2022-08-03T14:04:26Z&se=2022-08-03T22:04:26Z&skoid=c485ae09-a9af-40...

Blob SAS URL

https://nightlyconnxacuf7m5inn.blob.core.windows.net/diag?sp=rl&st=2022-08-03T14...

Copy to clipboard

2. Get a snapshot from the Pexip Infinity Management Node:

- a. On the Pexip Infinity Administrator interface, go to **Utilities > Diagnostic Snapshot**.
- b. To download all available data, select **Download full snapshot**.

or

To download a subset of the snapshot, containing a specified time period:

- i. Type in, or adjust the sliders to the start and end times (in terms of how many hours ago from the current time) of the diagnostic data to be downloaded. The number of hours available will vary depending on the logs available on the system.
- ii. If requested by your Pexip authorized support representative, select **Include diagnostic metrics from all Conferencing Nodes**.
If this is selected, then metrics from the Management Node and all Conferencing Nodes are included with the snapshot.
If this is not selected, then metrics from the Management Node only are included with the snapshot.
- iii. Select **Download limited duration snapshot**.

Wait while the snapshot file is prepared — do not navigate away from the page until the file has been generated.

- c. Follow your browser's prompts to save the file.

3. Upload the Azure logs and the Management Node snapshot to Pexportal:

- a. Go to pexportal.pex.me and sign in.
- b. You have to upload the Azure logs and the Pexip Infinity snapshot as two separate steps. First, upload the Azure logs:
 - i. Select **Upload Files**.
 - ii. Fill in the page details:



Ticket number	Enter your support ticket number.
Azure SAS URL	Paste in here the Blob SAS URL from the Azure portal and select Fetch dates .
Select a file to upload	Ignore this button when uploading the Azure logs.

]pexip[Pexportal

Home Upload files Tools ▾

Upload a new file

Use this form to upload files related to a support ticket.

Ticket number	<input type="text" value="1234"/>
Partner name	<input type="text" value="Pexip"/>
End customer name	<input type="text" value="Customer name"/>
Azure SAS URL	<input type="text" value="https://nightlyconnxacuf7m5inn.blob.core.windows.net/diag?sp=r&st=2022"/> Fetch dates
Start date	<input type="text" value="15/07/2022"/> <input type="text" value="End date"/> <input type="text" value="04/08/2022"/>
Select a file to upload	Choose file
<div>  I'm not a robot  reCAPTCHA Privacy - Terms </div>	
Upload	

- iii. Select **Upload**.

You see a message that the Azure logs are queued for download.

- c. Next, upload the Pexip Infinity snapshot:
 - i. Select **Upload Files**.
 - ii. Fill in the page details:


Ticket number	Enter your support ticket number.
Azure SAS URL	Leave blank
Select a file to upload	Select Choose file and select the diagnostic snapshot file you downloaded from the Management Node.

]pexip[Pexportal

Home Upload files Tools ▾

Upload a new file

Use this form to upload files related to a support ticket.

Ticket number	<input type="text" value="1234"/>
Partner name	<input type="text" value="Pexip"/>
End customer name	<input type="text" value="Customer name"/>
Azure SAS URL	<input type="text"/> <button>Fetch dates</button>
Select a file to upload	<input type="text" value="diagnostic_snapshot_10-115-104-2-mgr_22_08_04_15_40_14.tgz"/>
<div><div><input checked="" type="checkbox"/> I'm not a robot</div><div> reCAPTCHA Privacy Terms</div></div>	
<input type="button" value="Upload"/>	

iii. Select Upload.

You see a message that the that the snapshot file has been uploaded.

You can now exit Pexportal.

Data stored/processed by the Teams Connector

Information related to the processing of calls into Microsoft Teams meetings is logged and stored by the Teams Connector. This includes, but is not limited to, Conferencing Node IP address information and FQDNs, VTC display names / caller aliases, call IDs, Teams conference IDs, Teams meeting participant names / aliases, application IDs, trusted call status, participant lobby status, and participant presentation status.

These logs are stored in an Azure Storage account (owned by the customer) and are rotated daily with expiry after 90 days.

In addition to the logged data:

- The Teams Connector stores the TLS certificate used to verify connections to the Teams Connector.
- All call signaling and media between Conferencing Nodes and the Microsoft Teams backend passes through the Teams Connector.

Discovering your Azure tenant ID

To retrieve your tenant ID from the Azure portal:

1. Select **Azure Active Directory**.
2. Under **Manage**, select **Properties**.
3. The tenant ID is shown in the **Directory ID** field. You can use the **Click to copy** option to copy it.

You can also check your tenant ID at <https://www.whatismytenantid.com/>.

Viewing your tenant's app registrations

Your Pexip CVI application needs to be granted permission to enable access to Microsoft Teams in an Office 365 tenant, as described in [Authorize Pexip CVI application to join Teams meetings](#).

You can check the status of your apps by viewing your app registrations and your list of enterprise applications.

Viewing current app registrations

To see the current app registrations for your Azure tenant:

1. Go to the Azure portal (<https://aad.portal.azure.com>) and sign in.
2. Go to **Azure Active Directory > App Registrations**.
3. Search for the prefix you used when deploying your Teams Connector, and it will list your app registrations as shown below (your app names will be different).

Home > Pexip AS

Pexip AS | App registrations

Azure Active Directory

Overview
Preview features
Diagnose and solve problems

Manage
Users
Groups
External Identities
Roles and administrators
Administrative units
Enterprise applications
Devices
App registrations
Identity Governance

+ New registration | Endpoints | Troubleshooting | Refresh | Download | Preview features

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azu continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need Microsoft Authentication Library (MSAL) and Microsoft Graph. [Learn more](#)

All applications | **Owned applications** | Deleted applications

Search: pexip | Application (client) ID starts with

2 applications found

Display name	Application (client) ID	Created on
pp pexip-TeamsConn-eu Pexip Teams Connector API	4b3cec77-db25-4646-a4c1-bda6...	12/10/2021
DE pexipTeamsConn	bddb493d-d8e8-4921-b945-026...	12/10/2021

Note that if you have deployed the Teams Connector several times, old app registrations will also show — even if you have deleted the Azure Bot resource group. If required, you can delete these old registrations.

4. To check if consent has been granted in your Tenant, select the CVI app registration from the list (the one named "<prefix>TeamsConn").

If it shows **Create Service Principal** (as shown below) then consent has not been granted in your own Azure AD. (Note that this is expected if Teams is in a different Azure AD from your Azure Bot, however in most enterprises it will be in the same Azure AD.)

Home > Pexip AS - App registrations > pexipTeamsConn

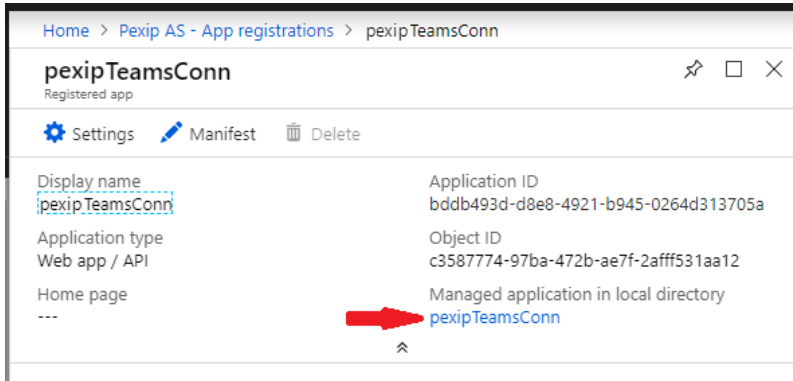
pexipTeamsConn
Registered app

Settings | Manifest | Delete

Display name	Application ID
pexipTeamsConn	bddb493d-d8e8-4921-b945-0264d313705a
Application type	Object ID
Web app / API	c3587774-97ba-472b-ae7f-2aff531aa12
Home page	Managed application in local directory
---	Create Service Principal

If consent has been granted in your tenant, the registration will show a reference to the enterprise app ("pexipTeamsConn" in the

example below).

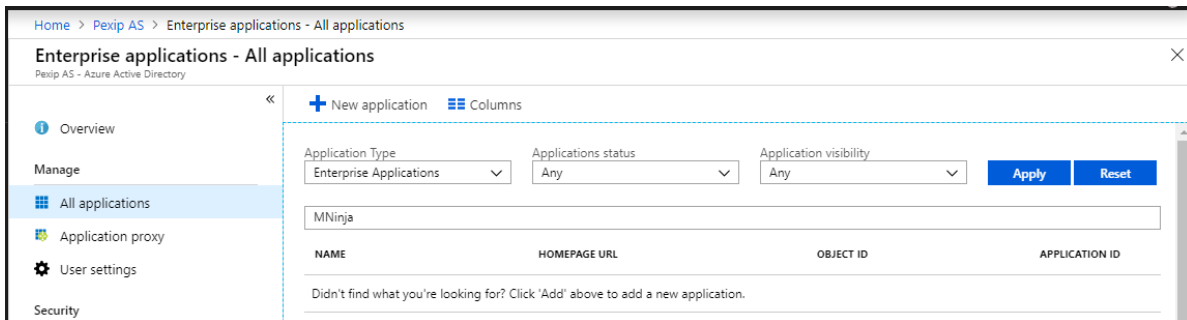


View authorized enterprise apps

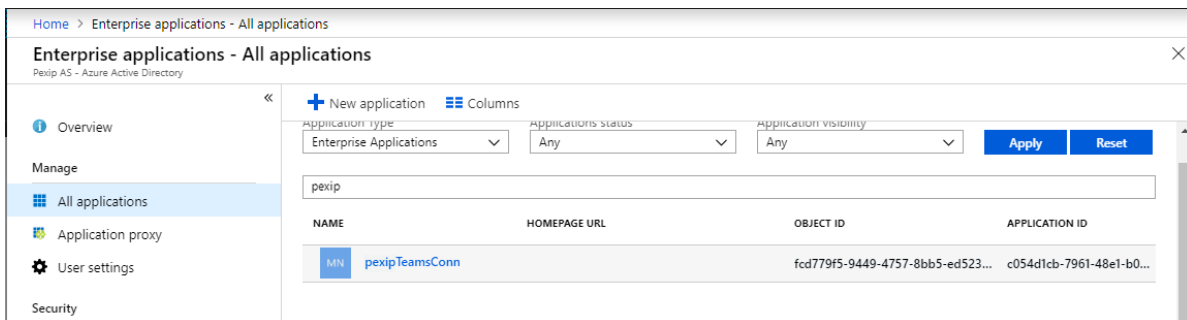
To view authorized enterprise apps i.e. where consent has been granted:

1. From the Azure portal go to **Azure Active Directory** > **Enterprise Applications**.
2. Search for the prefix you used when deploying your Teams Connector.

In the example screenshot below there are currently no apps where consent has been granted:



If the app has had consent granted, you will see the app as shown below (your app name will be different):



3. If your Teams Connector CVI app is not shown, you must provide consent to it as described in [Authorize Pexip CVI applications](#). Your app will then appear in the list of enterprise applications as shown in the second example screenshot above. Also when you view the app via **Azure Active Directory** > **App Registrations**, each registration will show a reference to the enterprise app.

Azure notification: Denial of service attack detected

You may receive notifications from Azure that it has detected potential outgoing denial-of-service attacks from your Teams Connector. They take the form:

Outgoing denial-of-service:

Azure Security Center detected unusual network activity originating from the virtual machine providers <component list>. An

unusually large number of connections were made from this machine. This anomalous activity may indicate that your virtual machine was compromised and is now engaged in denial-of-service (DoS) attacks against external endpoints.

This occurs when Azure detects the media traffic from your Teams Connector as potential UDP flooding. These messages can be safely ignored as the Teams Connector is behaving correctly, and Azure has not blocked your media connection.

Azure is developing an "Alert suppression" feature that in the future will allow you to disable these alerts for your subscription.