Integrating Microsoft Teams with Pexip Infinity

Deployment Guide

Software Version 20

Document Version 20.a

December 2018

pexip

Contents

Integration overview

Integrating Microsoft Teams with Pexip Infinity provides any-to-any video interoperability with Microsoft Teams.

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 / RTMP)

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

The key features of a Microsoft Teams integration are:

- Native VTC network resiliency
- Bi-directional content sharing between VTCs and Microsoft Teams via Video Based Screen Sharing (VbSS)
- Native scheduling via Outlook and Microsoft Teams client
- No need for Click to Run version (C2R) of Office 2016
- 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.

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 meeting ID for the relevant Microsoft meeting platform.

Deployment environments and options

This section explains how you can deploy the Pexip Teams Connector and the Pexip Infinity platform.

Pexip Teams Connector

The Pexip Teams Connector must be deployed in Microsoft Azure. The Teams Connector handles all Teams communications and meeting requests from the Pexip Infinity platform and passes them on to the Microsoft Teams environment.

The dedicated Pexip application ensures control and ownership for organizations with stringent regulatory compliance requirements.

Pexip Infinity platform

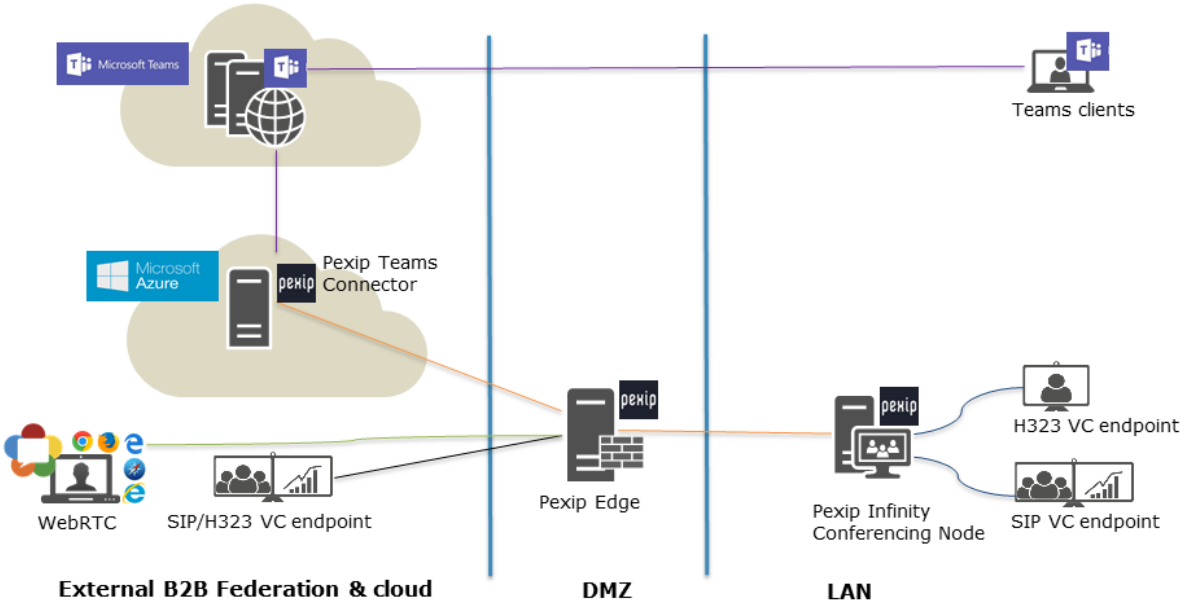
The Pexip Infinity platform can be deployed in any of its supported environments such as on-premises or in a public or hybrid cloud (which would typically be Microsoft Azure when integrating with Microsoft Teams).

The only requirements are that:

- You have one or more publicly-reachable Conferencing Nodes (either Proxying Edge Nodes or Transcoding Conferencing Nodes).
- Those Conferencing Nodes have certificates installed that have been signed by an external certificate authority (CA).

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.

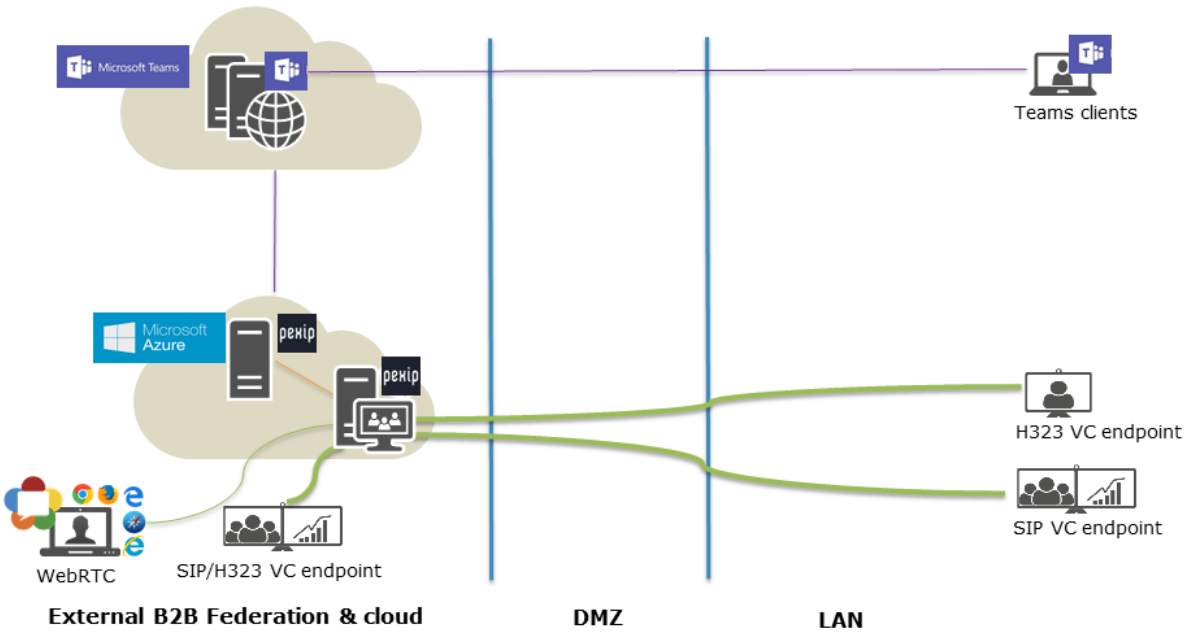


Pexip Teams Connector deployed in Azure and Pexip Infinity platform deployed on-premises

Cloud-hosted deployment

The Pexip Infinity platform can be deployed in a dedicated public or hybrid cloud. You could use any supported cloud service but you would typically deploy your Conferencing Nodes in Microsoft Azure alongside your Pexip Teams Connector. In this example deployment, external endpoints and federated systems, as well as on-premises devices can all connect to Teams conferences via the cloud-hosted Pexip Infinity nodes.

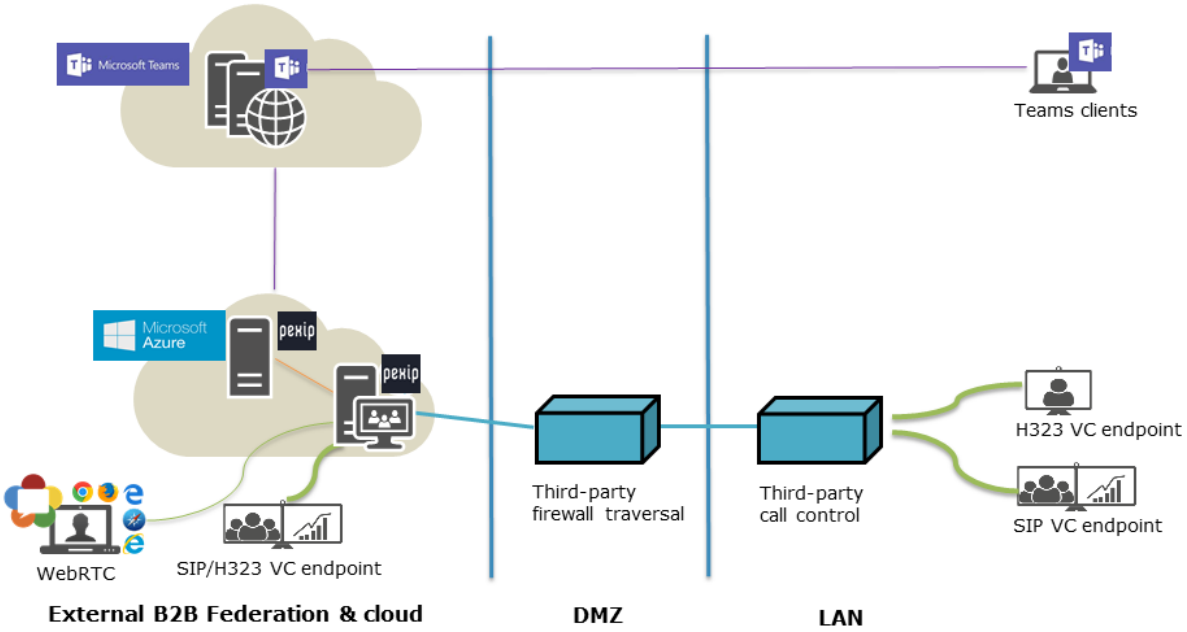
You run the Pexip platform from within your own cloud subscription, which means you have full control of your dedicated environment and network configuration.



Pexip Teams Connector and the Pexip Infinity platform deployed in Azure

Including third-party call control

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

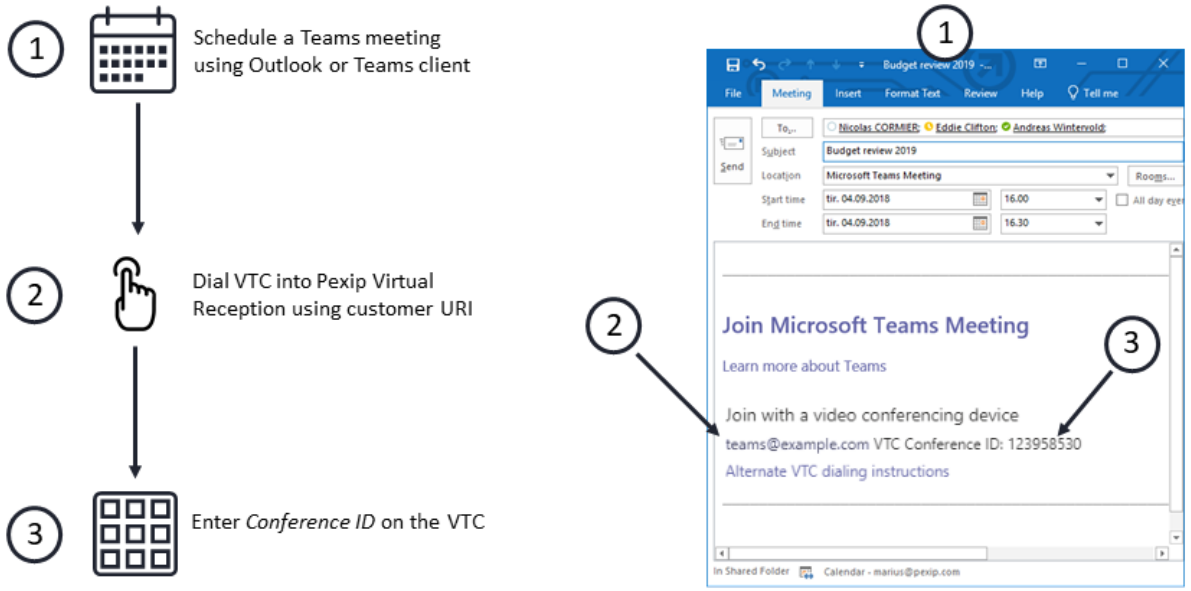


Pexip Teams Connector and the Pexip Infinity platform deployed in Azure with on-prem third-party call control

User experience when joining a 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 Meeting 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 Infinity Connect clients can also join via direct dial or via a Virtual Reception.

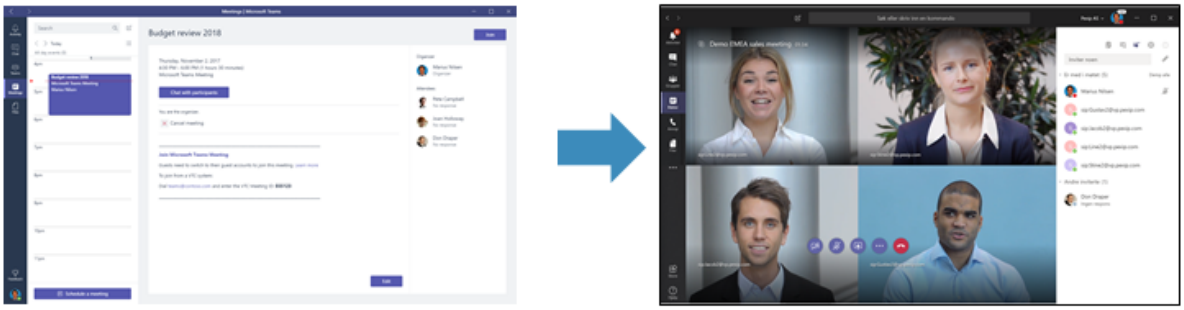
Pexip Infinity's deployment model allows the use of a customer-specific domain such as teams@example.com for dialing the Teams Virtual Reception.



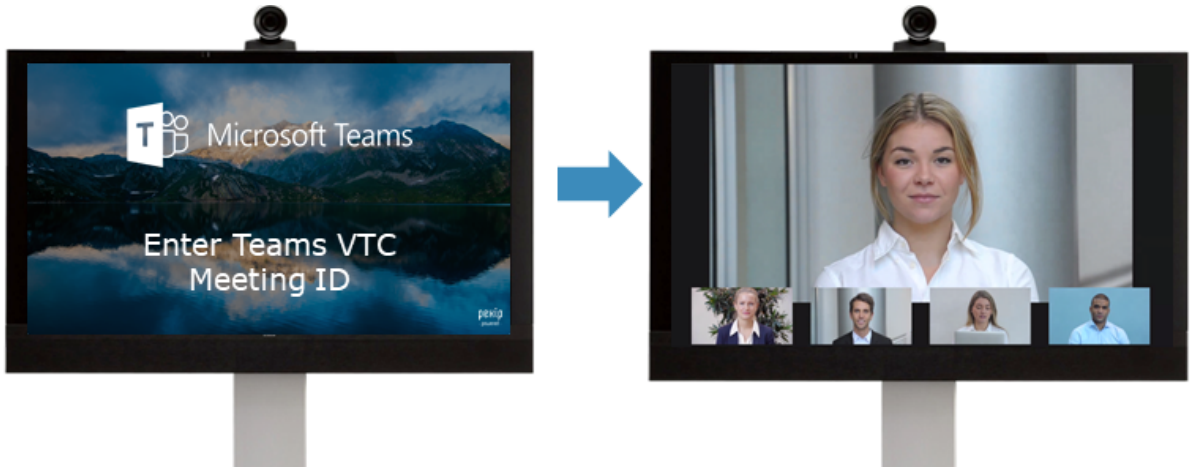
Alternative VTC dialing instructions can be provided that are customized to the company network and workflow such as:

- 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. 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



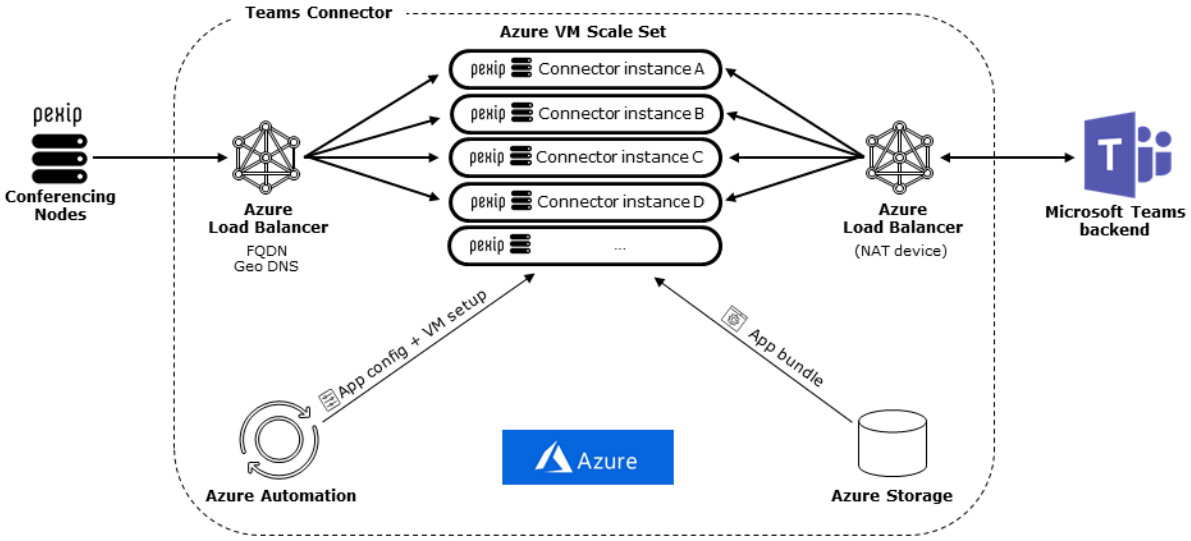
Join workflow from a third-party VTC system into a Microsoft Teams meeting (Pexip's standard 1+7 layout)

Installing and configuring the Teams Connector in Azure

The Pexip Teams Connector must be deployed in Microsoft Azure. The Teams Connector handles all Teams communications and meeting requests from the Pexip Infinity platform and passes them on to the Microsoft Teams environment.

Architecture overview

The following diagram shows the Teams Connector elements that are deployed in Azure, and how they interact with the Pexip Infinity platform and Microsoft Teams. You do not have to set up these Azure components individually — they are all created as part of the deployment process that is described below.



Note that:

- The Teams Connector must be deployed in Microsoft Azure. The Azure Virtual Machine scale set (VMSS) allows the Pexip application to run across many identical VMs, with automatic scaling of resources and load balancing of traffic.
- Azure virtual machine scale sets (VMSS) let you create and manage a group of identical, load balanced VMs. Azure virtual machine scale sets provide the management capabilities for applications that run across many VMs, automatic scaling of resources, and load balancing of traffic.
- The Pexip Infinity platform can be installed in any supported platform (including Microsoft Azure).
- The Pexip Conferencing Nodes:
  - can be Transcoding Conferencing Nodes or Proxying Edge Nodes, providing they have a publicly-reachable address
  - must have TLS certificates installed that have been signed by an external trusted CA (certificate authority)
  - can have static NAT and/or dual network interfaces, as the Teams Connector is treated as a lineside connection.

Preparation, prerequisites and capacity planning

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

Capacity planning

Contact your Pexip authorized support representative to discuss your call capacity requirements, and how many Teams Connector instances are required.

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.

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.

Azure regions with Fs series instance type support

This is the list of Azure regions that support Fs series instances and Automation as of September 2018.

Theater	Region	RegionName
US		
	East US 2	eastus2
	South Central US	southcentralus
	West US2	westcentralus
Canada		
	Canada Central	canadacentral
Europe		
	North Europe	northeurope
	West Europe	westeurope
UK		
	UK South	uksouth
APAC		
	Southeast Asia	southeastasia
Australia		
	Australia Southeast	australiasoutheast
India		
	Central India	centralindia
Japan		
	Japan East	japaneast
Korea		
	Korea Central	koreacentral

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-AzureRmResourceProvider -ProviderNamespace
Microsoft.Automation).ResourceTypes | Where-Object ResourceTypeName -eq
automationAccounts).Locations;
  $vmLocations = ((Get-AzureRmResourceProvider -ProviderNamespace
Microsoft.Compute).ResourceTypes | Where-Object ResourceTypeName -eq
locations/virtualMachines).Locations;
  $vmInstanceLocations = $vmLocations | ?{ 'Standard_F4s' -in @(Get-AzureRmVMSize -
```

```
Location $_ | %{ $_.Name }) };
  Get-AzureRmLocation | ?{ ($_.DisplayName -in $automationLocations) -and
($_.DisplayName -in $vmInstanceLocations) }
) | Select-Object -Property DisplayName,Location
```

- 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 recommended 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 days for the quota increase request to be processed. For more information see https://docs.microsoft.com/en-us/azure/azure-subscription-service-limits.

Install 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.
- 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.

Obtain and prepare 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).

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.

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 is what you will use as the value of **$PxTeamsConnFqdn** in the variables initialization script.
   - The certificate's subject name must match 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 a different domain to your Teams and Pexip Infinity deployments.
   - 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.
   - It can be a wildcard certificate.
2. Request a certificate for that name and generate the certificate in PFX format. Any intermediate certificates must also be in the PFX file.

You will use this certificate when installing the Teams Connector as described below.

Ensure 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 below) 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.

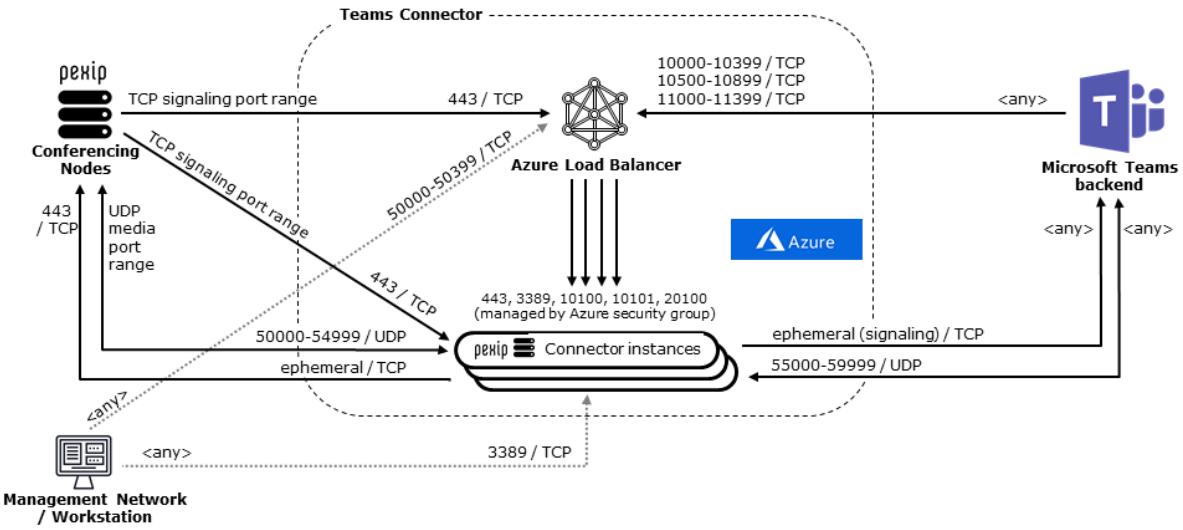
See Certificate and DNS examples for a Microsoft Teams integration for more information and examples about certificates, DNS records and using a "pool name" for Conferencing Nodes.

Review firewall ports for the Teams Connector

When deploying the Pexip Teams Connector, the following ports have to be allowed through any firewalls which carry traffic between the Teams Connector components and Microsoft Teams (O365) / Conferencing Nodes.

Source address	Source port	Destination address	Destination port	Protocol	Notes
Conferencing Nodes					
Conferencing Nodes	33000–39999*	Teams Connector load balancer	443	TCP	Signaling
		Teams Connector instance			
Conferencing Nodes	40000–49999*	Teams Connector instance	50000-54999	UDP	Call media
Teams Connector components					
Teams Connector instance	ephemeral	Microsoft Teams (O365)	<any>	TCP	Signaling
Teams Connector instance	ephemeral	Conferencing Nodes	443	TCP	Signaling
Teams Connector instance	50000-54999	Conferencing Nodes	40000–49999*	UDP	Call media
Teams Connector instance	55000-59999	Microsoft Teams (O365)	<any>	UDP	Call media
Microsoft Teams (O365)					
Microsoft Teams (O365)	<any>	Teams Connector load balancer	10000-10399	TCP	Signaling
			10500-10899		
			11000-11399		
Microsoft Teams (O365)	<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		

* The media and signaling port ranges on Conferencing Nodes are configurable.



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, 10101 and 20100) — these ports do not need to be opened between the Conferencing Nodes / Microsoft Teams and the Teams Connector.
- 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 (so that you can add addresses and allow it later if required).

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	...
110	Teams-signalling-trusted-endpoint	10101	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	...

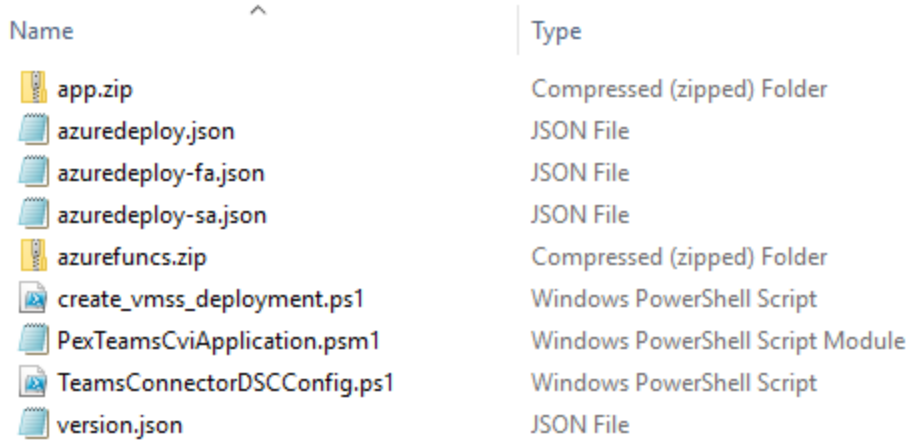
Outbound security rules

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	...

Download the Teams Connector application software files

First, you must download the Pexip Teams Connector application to the administrator PC.

1. Download the **Teams Connector ZIP** file (Pexip_Infinity_Connector_For_Microsoft_Teams_v20_<build>.zip) from the Pexip support site.
   Ensure that the Teams Connector version you download is the same version as your Pexip Infinity deployment.
2. Extract the files to a folder on your administrator PC.
3. Verify that the ZIP file is extracted and that you see the following files:



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

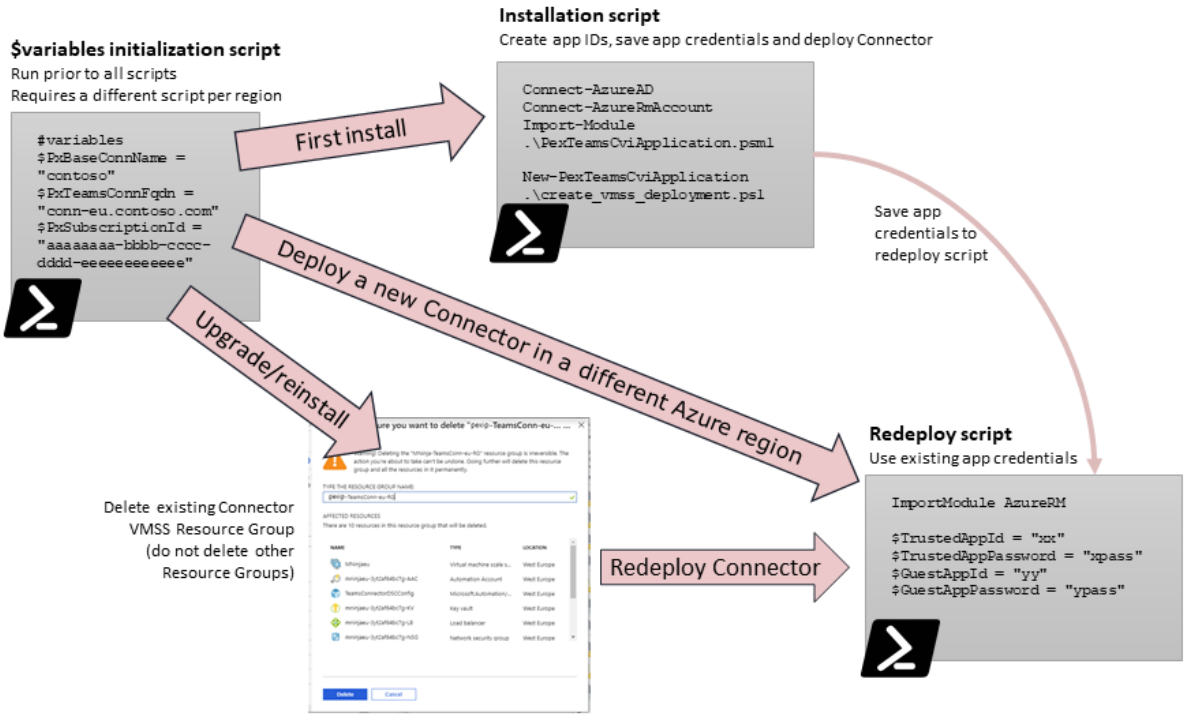
Installing the Teams Connector application into Azure

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. Run the installation script.
3. Save the App IDs and passwords output from the installation script into the redeploy script.
4. Update DNS with the Teams Connector's name and IP address.
5. Authorize the Pexip CVI applications to join Teams meetings.
6. Authorize the Trusted app to bypass the Teams lobby and configure dialing instructions.

If you subsequently need to redeploy or upgrade your Teams Connector, or deploy another Teams Connector in a different Azure region, you need to follow a different process as described in Maintaining your Teams Connector deployment.

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 Azure AD and Office 365 can be found at https://docs.microsoft.com/en-us/office365/enterprise/powershell/connect-to-office-365-powershell.

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 modify the values of any variables if required). This will make it easier if you have to abort and restart the installation for any reason, or if you have to redeploy or upgrade your Teams Connector in the future.

You must replace the example values set in the variables with the real values for your deployment.

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 next 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 cannot contain dashes, spaces or other non a-z0-9 chars.
$PxBaseConnName = "yourname" # replace with your company name

# Freetext region shortname (no dashes, only use a-z) to separate regional deployments
$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
$PxSubscriptionId = "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"

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

# Username for the Windows VM accounts
$PxWinAdminUser = "pexadmin"

# Password for the Windows VM accounts (can be set with Get-Credential if desired)
$PxWinAdminPassword = "ReplaceThisPassword!" # Password for Windows account

# Number of Teams Connector VMs
$PxTeamsConnInstanceCount = "3"

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

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

# Enable incident reporting
$PxTeamsConnIncidentReporting = $true

# Enable VMSS disk encryption
# This requires Disk Encryption to be registered on the subscription first
# https://blogs.msdn.microsoft.com/azuresecurity/2017/09/28/announcing-azure-disk-
encryption-preview-for-virtual-machine-scale-sets/
# Verify if this is enabled by running:
# Get-AzureRmProviderFeature -ProviderNamespace "Microsoft.Compute" -FeatureName
"UnifiedDiskEncryption"
$PxTeamsConnDiskEncryption = $false

# 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"

# Management networks - used for RDP access and admin consent
# 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.0/24  Management subnet
# $PxMgmtSrcAddrPrefixes = @( "x.x.x.x", "y.y.y.y", "z.z.z.0/24" )
$PxMgmtSrcAddrPrefixes = @()

# Pexip public facing Conferencing / Edge node IP addresses
# If not specified (default) - HTTPS access is always enabled
```

```
# If specifying IPs, then in addition to the Pexip node IP addresses you MUST also
include
# the external IP address of the workstation / management network that will be used to
# provide consent for the Teams Connector apps (otherwise 443 will be blocked)
#
# Example Pexip Edge nodes public IP source ports:
# 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)
#
# Example (specifying Pexip edge nodes and Management networks defined above):
# $PxNodesSourceAddressPrefixes = @( "a.a.a.a", "c.c.c.0/28" ) + $PxMgmtSrcAddrPrefixes
$PxNodesSourceAddressPrefixes = @()
```

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	This is a prefix used when naming all Teams Connector resources. We recommend using your own company name. **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. 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.	
$PxVmssRegion	A short (we recommend 2-4 characters) name 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. It cannot contain dashes, spaces or other non a-z0-9 characters.	✓
$PxTeamsConnFqdn	The hostname of the Teams Connector, for example "**pexip-TeamsConn-eu.teams.example.com**". This name must match the subject name in the PFX certificate specified in **$PxPfxCertFileName**. If you are installing multiple Teams Connectors in different regions you must use a different hostname for each region. This is also the name you will configure in Pexip Infinity (**Call Control > Microsoft Teams Connectors > Address Of Teams Connector**) later in the process.	✓
$PxNodeFqdns	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. The name(s) specified here must exist in the certificate presented by those Conferencing Nodes. See Ensure Conferencing Nodes have suitable certificates for more information.	✓ (typically)
$PxSubscriptionId	The Azure Subscription ID for your Teams Connector installation and botchannel registrations. This takes the format "**aaaaaaaa-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.	✓

Variable name	Description and example usage	Regional
$PxWinAdminUser	The account username for the Windows VMs that will be created in Azure. You may need this for RDP login when troubleshooting. See https://docs.microsoft.com/en-us/azure/virtual-machines/windows/faq#what-are-the-username-requirements-when-creating-a-vm for username requirements.	

$PxWinAdminPassword	The account password for the Windows Teams Connector VMs that will be created in Azure. Alternatively you can use the Get-Credential cmdlet to set the password.
	
	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 "-"
	• be between 12 and 72 characters long
	• not include reserved words or unsupported characters.	

$PxTeamsConnInstanceCount	The number of Teams Connector instances (VMs) to deploy, for example "**3**". You can easily modify the number of Teams Connector instances via the Azure portal after installation of the Teams Connector, to reflect changing capacity requirements.	
$PxTeamsConnResourceGroupName	The regional resource group name. We recommend setting this variable to a concatenated combination of the first two variables plus some additional text, for example: **$PxTeamsConnResourceGroupName = "$($PxBaseConnName)-TeamsConn-$($PxVmssRegion)-RG"** which, using our examples above, would generate a resource group name of **pexip-TeamsConn-eu-RG**.	
$PxTeamsConnStaticResourceGroupName	The static regional resource group name. This is used in the scripts for retrieving name and address information for the load balancer. This should follow the same pattern as the previous variable ($PxTeamsConnResourceGroupName), and be set to a concatenated combination of the first two variables plus some additional text, for example: **$PxTeamsConnStaticResourceGroupName = "$($PxBaseConnName)-TeamsConn-$($PxVmssRegion)-static-RG"** which, using our examples above, would generate a static resource group name of **pexip-TeamsConn-eu-static-RG**.	
$PxTeamsConnIncidentReporting	When this feature is enabled, incident reports are sent automatically to a secure web server owned and managed by Pexip. The options are **$true** to enable incident reporting or **$false** to disable reporting. We recommend keeping this enabled.	
$PxTeamsConnDiskEncryption	Controls whether VMSS (virtual machine scale sets) disk encryption is enabled. If you want to enable this you must first register Disk Encryption on the Azure subscription. See https://blogs.msdn.microsoft.com/azuresecurity/2017/09/28/announcing-azure-disk-encryption-preview-for-virtual-machine-scale-sets/ for information about how to do this. You can verify if it is registered by running: `Get-AzureRmProviderFeature -ProviderNamespace "Microsoft.Compute" -FeatureName "UnifiedDiskEncryption"` The options are **$true** to enable encryption or **$false** to disable encryption. When enabled, the keys are stored in an Azure Key Vault.	

Variable name	Description and example usage	Regional
$PxPfxCertFileName	The filename of the PFX certificate file to upload to the Teams Connector. See Install Teams Connector and Conferencing Node certificates for more information.	✓

$PxMgmtSrcAddrPrefixes	Specifies the IP addresses of any management workstations / networks that may be required to administer the Teams Connector instances over RDP.
	
	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. Note that you should not perform any security scans from these addresses.
	
	For example to allow RDP access from:
	
	`x.x.x.x  Management IP address #1`
	`y.y.y.y  Management IP address #2`
	`z.z.z.0/24  Management subnet`
	
	you would specify **$PxMgmtSrcAddrPrefixes = @("x.x.x.x", "y.y.y.y", "z.z.z.0/24")**
	If no addresses are specified i.e. **$PxMgmtSrcAddrPrefixes = @( )** then all RDP access is blocked.	

$PxNodesSourceAddressPrefixes	Specifies the IP addresses of the Conferencing Nodes (typically Proxying Edge Nodes) that can communicate with the Teams Connector instances over port 443 (https).
	
	This is used to populate the Azure Network Security Group "MCU-signalling-endpoint" rule.
	
	ℹ You must also include the external IP address (as seen from the Teams Connector's perspective) of the workstation / management network that will be used to provide consent for the Teams Connector apps to access Microsoft Teams in the Office 365 tenant.
	
	For example, if these are the public addresses of your Pexip Conferencing Nodes:
	
	`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)`
	
	you would specify:
	$PxNodesSourceAddressPrefixes = @("a.a.a.a", "c.c.c.0/28") + $PxMgmtSrcAddrPrefixes
	which sets the variable to the addresses of your Conferencing Nodes **plus** the IP addresses of any management workstations / networks (as specified in the previous variable).
	If no addresses are specified i.e. **$PxNodesSourceAddressPrefixes = @( )** then anything can connect via https to the Teams Connector instances.
		✓ (typically)

Installation commands to connect to AzureAD, AzureRm and deploy the Teams Connector

Here are the PowerShell commands to connect to AzureAD, AzureRm, set up Trusted and Guest App IDs, and then deploy the Teams Connector. The purpose of each command is explained as a comment within the script.

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

- 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. Note that 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.
- If you are connecting to Azure AD/Rm from your Windows PC for the first time, you must first run the following PowerShell commands (as Administrator):
  ```
  Install-Module -Name AzureAD
  Install-Module -Name AzureRM -AllowClobber
  ```
- You must have the "Contributor" role in the Azure subscription used for the Teams Connector.

- The AzureAD login you use must be a user in the tenant, not a Windows Live ID account.
- The following script is referred to as the installation script. This script only needs to be run once (per Azure subscription).
  If you need to upgrade or redeploy your Teams Connector, or deploy a new Teams Connector in a different Azure region you should use the redeploy script.
- Note that the **create_vmss_deployment.ps1** command in the script that creates the Teams Connector VMs can take up to 30 minutes to complete.

```
# Connect to PowerShell AzureAD, AzureRm and import Pexip CVI module
# Azure AD commands
# Connect to AzureAD
# If AAD/365 admin is not the same as Azure Resource Manager admin,
# 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
#
Connect-AzureAD

# Set execution policy for the current PowerShell process
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# Connect to Azure Resource Manager PowerShell (in same window to reuse variables)
# This step can be omitted if you only are running the AAD commands to create trusted
Apps
Import-Module AzureRM -MinimumVersion 6.0.0
Connect-AzureRmAccount

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

# Create two new CVI Applications, one Trusted, and one Guest
# Create Trusted App
$TrustedApp = New-PexTeamsCviApplication -AppDisplayName
"$($PxBaseConnName)TeamsConnTrusted" -ConnectorFqdn $PxTeamsConnFqdn -Confirm:$false
$TrustedAppId = $TrustedApp.AppId

# Create Trusted App Password
$TrustedAppPassword = ($TrustedApp | New-PexTeamsCviApplicationPasswordCredential -
KeyIdentifier default).Value

# Create Guest App
$GuestApp = New-PexTeamsCviApplication -AppDisplayName "$($PxBaseConnName)TeamsConnGuest"
-ConnectorFqdn $PxTeamsConnFqdn -Confirm:$false
$GuestAppId = $GuestApp.AppId

# Create Guest App Password
$GuestAppPassword = ($GuestApp | New-PexTeamsCviApplicationPasswordCredential -
KeyIdentifier default).Value

Write-Host
Write-Host
Write-Host "`n----------------------------------------`n"
Write-Host
Write-Host "### App ID and credentials MUST be saved in the redeploy script ###"
Write-Host
Write-Host "`$TrustedAppId = `"$($TrustedAppId)`""
Write-Host "`$TrustedAppPassword = `"$($TrustedAppPassword)`""
```

```
Write-Host "`$GuestAppId = `"$($GuestAppId)`""
Write-Host "`$GuestAppPassword = `"$($GuestAppPassword)`""
Write-Host
Write-Host "`n----------------------------------------`n"
Write-Host
Write-Host
```

Azure RM commands

```
# Change context to the Pexip Subscription and set the Trust/Guest credentials
Set-AzureRmContext -SubscriptionId $PxSubscriptionId

$TrustedAppSecurePassword = ConvertTo-SecureString -AsPlainText $TrustedAppPassword -
Force
$TrustedAppCred = New-Object System.Management.Automation.PSCredential -ArgumentList
$TrustedAppId,$TrustedAppSecurePassword

$GuestAppSecurePassword = ConvertTo-SecureString -AsPlainText $GuestAppPassword -Force
$GuestAppCred = New-Object System.Management.Automation.PSCredential -ArgumentList
$GuestAppId,$GuestAppSecurePassword

# Bot channel registration
# The Bot Channel registration is used globally, and only needs to be created once (in
any of your regions)

# Bot channel Resource group creation (in your main region)
New-AzureRmResourceGroup -Location $PxAzureLocation -ResourceGroupName
"$($PxBaseConnName)-TeamsBotChan-RG"

# Bot channel registrations for the trusted and the guest AppID
# Create trusted bot
Register-PexTeamsCviApplicationBot -SubscriptionId $PxSubscriptionId -ResourceGroupName
"$($PxBaseConnName)-TeamsBotChan-RG" -BotName "$($PxBaseConnName)-Trusted-TeamsBot" -
AppId $TrustedAppId -Confirm:$false

# Create guest bot
Register-PexTeamsCviApplicationBot -SubscriptionId $PxSubscriptionId -ResourceGroupName
"$($PxBaseConnName)-TeamsBotChan-RG" -BotName "$($PxBaseConnName)-Guest-TeamsBot" -AppId
$GuestAppId -Confirm:$false

# 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 variables above) for the Windows VM
$PxWinAdminSecurePassword = ConvertTo-SecureString -AsPlainText $PxWinAdminPassword -
Force
$PxWinAdminCred = New-Object System.Management.Automation.PSCredential -ArgumentList
$PxWinAdminUser,$PxWinAdminSecurePassword

# Optionally if you do not prefer to have a password set as a variable, use Get-
Credential
# $PxWinAdminCred = Get-Credential

# Create Resource group for Teams Connector Load Balancer (per region)
# This stores the public IP address of the Teams Connector Load Balancer
# so that the address can be re-used on upgrade or redeploy
New-AzureRmResourceGroup -Location $PxAzureLocation -ResourceGroupName
$PxTeamsConnStaticResourceGroupName -Force

# Create Resource group for Teams Connector VMSS (per region)
```

```
New-AzureRmResourceGroup -Location $PxAzureLocation -ResourceGroupName
$PxTeamsConnResourceGroupName

# Ensure required version of the Carbon Powershell module is used
(Get-Content TeamsConnectorDSCConfig.ps1).replace('Install-Module -Name Carbon -
AllowClobber', 'Install-Module -Name Carbon -RequiredVersion 2.6.0 -AllowClobber') | Set-
Content TeamsConnectorDSCConfig.ps1

# 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 -
TrustedAppCredential $TrustedAppCred -GuestAppCredential $GuestAppCred -
StaticResourcesResourceGroupName $PxTeamsConnStaticResourceGroupName -
PublicIPAddressResourceName "$($PxBaseConnName)-TeamsConn-$($PxVmssRegion)-PIP" -
IncidentReporting $PxTeamsConnIncidentReporting -Encryption $PxTeamsConnDiskEncryption -
RdpSourceAddressPrefixes $PxMgmtSrcAddrPrefixes -PexipSourceAddressPrefixes
$PxNodesSourceAddressPrefixes

# 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 AzureAD and
AzureRM)
#
# Setting subscription
Set-AzureRmContext -SubscriptionId $PxSubscriptionId

# Getting Network Security Group Resource ID
$nsgResId = (Get-AzureRmResource -ResourceGroupName $PxTeamsConnResourceGroupName -
ResourceType Microsoft.Network/networkSecurityGroups).ResourceId

# Getting Public IP details
$publicIpAddress = (Get-AzureRmPublicIpAddress -ResourceGroupName
$PxTeamsConnStaticResourceGroupName).IpAddress
$publicIpFqdn = (Get-AzureRmPublicIpAddress -ResourceGroupName
$PxTeamsConnStaticResourceGroupName).DnsSettings.Fqdn
# Getting Tenant Details
$tenant = Get-AzureADTenantDetail
$tenantDomain = ($tenant.VerifiedDomains | Where-Object { $_._Default -eq $True }).Name
# 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 CNAME from
your official hostname"
Write-Host "then the Office 365 admin must consent for the AppIds to join Teams Meetings"
Write-Host
Write-Host "1) Setup a DNS CNAME for $($PxTeamsConnFqdn) pointing to "
Write-Host "   $($publicIpFqdn)"
Write-Host
Write-Host "   When this is done, and you can confirm a DNS lookup of $($PxTeamsConnFqdn)
resolves to"
Write-Host "   your Public IP of the load balancer ($($publicIpAddress)) - you are ready
```

```
to proceed."
Write-host
Write-Host "2) Give consent to trusted and guest apps. Go to:
https://$($PxTeamsConnFqdn)/adminconsent"
Write-Host
Write-Host "   If Management Source Address prefixes and Pexip Conferencing Node IPs are
defined,"
Write-Host "   the administrator doing consent must come from one of these addresses."
Write-Host "   Pexip node prefixes: $($PxNodesSourceAddressPrefixes)"
Write-Host "   If your Office 365 admin comes from a different IP, add it to the Azure
Network Security Group (NSG)"
Write-Host
Write-Host "   NSG inbound rules can be found here:"
Write-Host "
https://portal.azure.com/#@${tenantDomain}/resource${nsgResId}/inboundSecurityRules"
Write-Host
Write-Host "   Add an additional inbound rule allowing 443/TCP from your Office 365
administrator's source IP to Any"
Write-Host
Write-Host "   This is not required if you had no Management Prefixes - you can access it
directly."
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 App IDs and credentials, similar to this:

   ```
   ### App ID and credentials MUST be saved in the secondary script ###"

   $TrustedAppId = "c054d1cb-7961-48e1-b004-389xxxx32"
   $TrustedAppPassword = "bXl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3JkbXl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3JkbXl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3Jk=="
   $GuestAppId = "18c0476d-ee99-4673-a6da-xxxxx"
   $GuestAppPassword = "eHl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3JkbXl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3JkbXl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3Jk=="
   ```

2. Copy the four output lines in red that define the App IDs and passwords and paste them into a copy of the redeploy script, replacing the existing lines that say:

   ```
   $TrustedAppId = ""
   $TrustedAppPassword = ""
   $GuestAppId = ""
   $GuestAppPassword = ""
   ```

   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 apps. Instead you will set the variables to the IDs and passwords of the apps 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 you upgrade or if you need to redeploy, or deploy in another region.

   It is critical that you update and store the redeploy script with the Trusted/Guest app ID and passwords to ensure that upgrades/redeploys can be done using the same app IDs.

Update DNS with Teams Connector name and IP address

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

You must update DNS before conducting the app consent.

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 AzureRM and then rerun the last section of the deployment script that prints out the details for you:

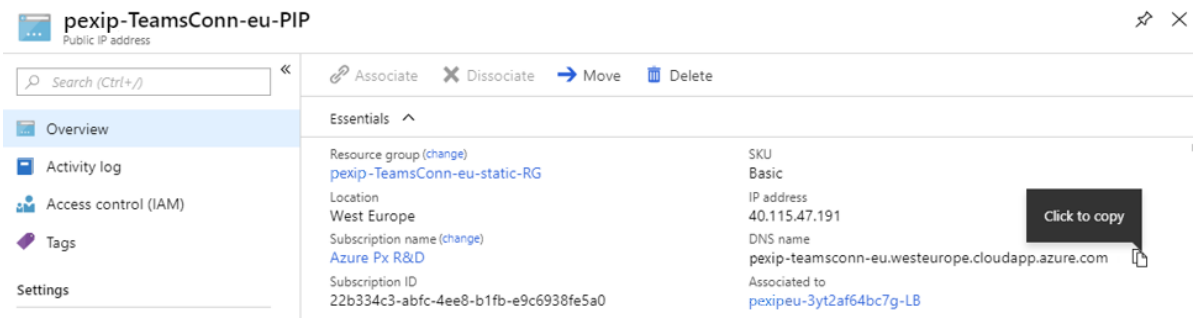
```
1) Setup a DNS CNAME for pexip-TeamsConn-eu.teams.example.com pointing to
   pexip-TeamsConn-eu.westeurope.cloudapp.azure.com

   When this is done, and you can confirm a DNS lookup of pexip-TeamsConn-eu.teams.example.com resolves to
   your Public IP of the load balancer 40.115.47.191 - you are ready to proceed.
```

Within DNS you need to set up a CNAME record that links the Teams Connector hostname/FQDN to the DNS name that was assigned by Azure to the load balancer. You do not have to set up any A records.

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 DNS name 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**.
3. Hover over the **DNS name** field and a "Click to copy" option appears. Click the option to copy the DNS name, for example **pexip-TeamsConn-eu.westeurope.cloudapp.azure.com**.



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

```
NAME                                    TYPE   VALUE
--------------------------------------  -----  ------------------------------------------------
pexip-TeamsConn-eu.teams.example.com.   CNAME  pexip-TeamsConn-eu.westeurope.cloudapp.azure.com.
```

You can confirm that the DNS record is working by performing a DNS lookup on the Name (**pexip-TeamsConn-eu.teams.example.com** in this example), which should resolve to the IP address of the load balancer (40.115.47.191 in this example).

You can now continue and provide consent to i.e. authorize the Pexip CVI apps.

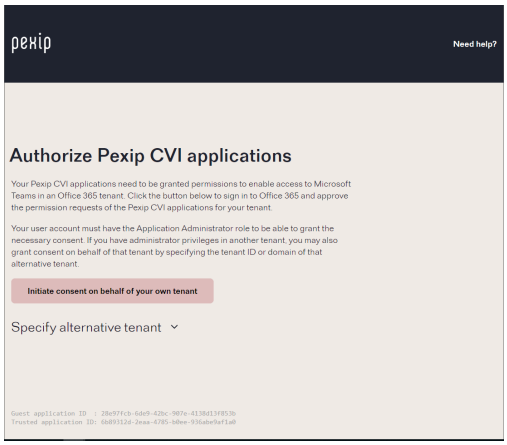
Authorize Pexip CVI applications to join Teams meetings

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

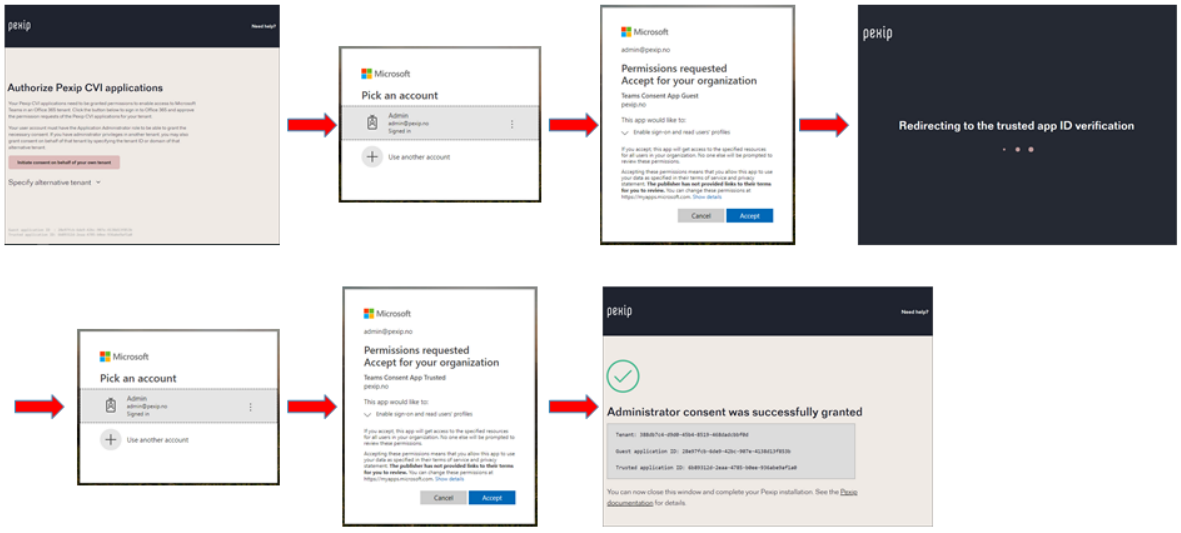
The end of the installation script produced instructions for the URL to visit to provide the necessary consent, for example:

```
2) Give consent to trusted and guest apps. Go to: https://pexip-TeamsConn-eu.teams.example.com/adminconsent
```

To provide consent to join Teams meetings:



1. 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 Trusted and Guest apps.
2. 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.
3. You are asked to confirm the account. This must be an account with the Application Administrator role, to be able to grant the necessary consent.
4. The permissions of the Pexip Teams Guest Connector are listed. The domain shown here is the domain where the App registration was created (this is the bot channel registration 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.
5. You are redirected to choose the account again, to sign the Pexip Teams Trusted Connector.
6. The same list of permissions has to be accepted for the Trusted Connector.
7. 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.

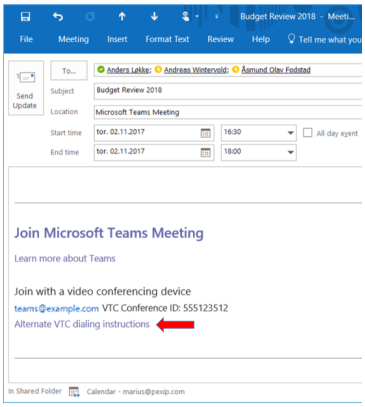


Authorize Trusted app to bypass Teams lobby and configure dialing instructions

There are two Pexip apps that route calls into Teams meetings, referred to as the **Trusted app** and the **Guest app**.

- The **Trusted app** is used to route calls from trusted participants, such as internal employees or calls placed from registered and authenticated devices. The **Trusted app** can then be assigned the ability to bypass the Teams lobby, meaning callers routed via the **Trusted app** are admitted directly into the Teams meeting.
- The **Guest app** is used to route calls from untrusted participants, such as external guests or calls placed from unknown devices. The **Guest app** is not assigned the Teams lobby bypass capabilities, meaning callers routed via the **Guest app** are held in the Teams lobby and have to be admitted into the meeting by an existing participant.
- Whether the **Trusted app** or the **Guest app** is used 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 use the **Trusted app** to route

the call into the Teams meeting. If the **Treat as trusted** option is not enabled on the rule, Pexip uses the **Guest app** to route the call into the Teams meeting. See Call Routing Rules for direct and indirect routing for more information.



- When a participant receives an invite to a Teams meeting, an "Alternate VTC dialing instructions" link to a webpage of alternate 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 Skype Online Powershell module:

1. Go to https://www.microsoft.com/en-us/download/details.aspx?id=39366 and download **SkypeOnlinePowerShell.Exe**.
2. Go to your download folder and run **SkypeOnlinePowerShell.Exe**.
   Note that the installation may fail if you do not have a compatible version of Microsoft Visual C++. The latest versions of Microsoft Visual C++ are available at https://support.microsoft.com/en-au/help/2977003/the-latest-supported-visual-c-downloads.
3. Agree to the terms and **Install** the module.
4. Start a PowerShell session and run the following commands, where **<tenant_name>** needs to be your onmicrosoft.com domain for your tenant:
   ```
   Import-Module SkypeOnlineConnector
   $sfbSession = New-CsOnlineSession -OverrideAdminDomain "<tenant_name>.onmicrosoft.com"
   Import-PSSession $sfbSession
   ```

Defining the Trusted app and Guest 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 **Trusted app** to bypass the Teams lobby
- specify the content of the alternative dialing instructions.

The command takes the form:

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

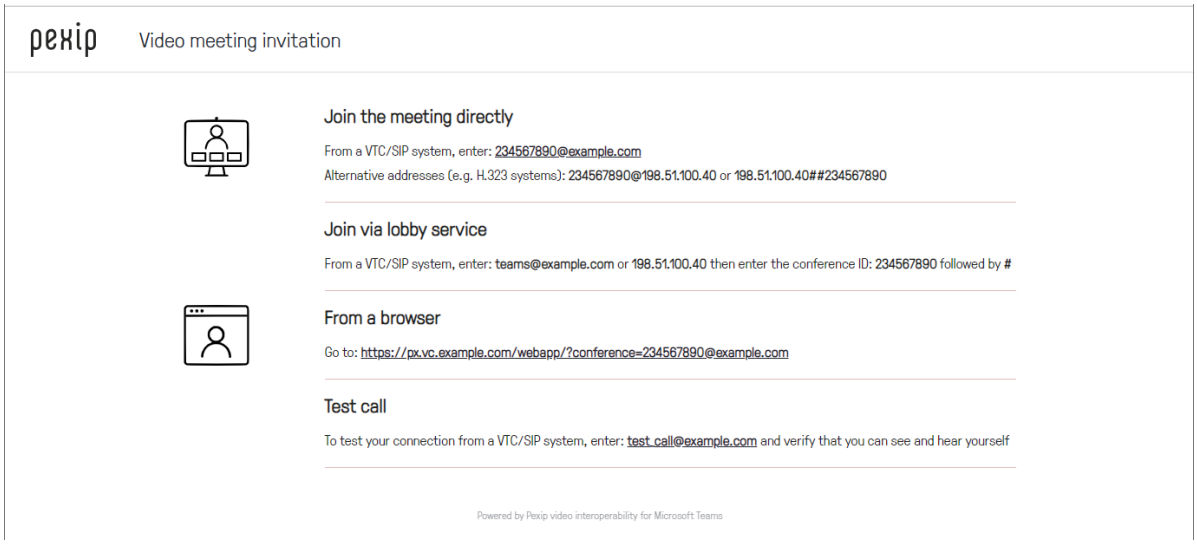
which contains the following parameters:

- **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 always the **Trusted app ID** and is used in conjunction with setting `AllowAppGuestJoinsAsAuthenticated $true` to allow the **Trusted app** to bypass the Teams lobby. Use the **Trusted 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`
  where **<node_address>** is the "pool name" of your Pexip Conferencing Nodes (it can also be the FQDN or IP address of an individual node).
  The **InstructionUri** parameters are:

Parameter	Mandatory	Description
conf	Yes	This must be set to **{ConfId}** 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 **d** parameter below. Note that **ivr@d** 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. 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 **conf** 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	The pubic-facing IP address of one of your Conferencing Nodes. You can only specify a single address. When specified, alternative dialing options by IP address (which may be required by some H.323 systems for example) are displayed on the webpage. If included, the IP address that you specify 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 used as the alias to dial. This uses Pexip Infinity's inbuilt Test Call Service; you must also ensure that you set up the associated alias e.g. **test_call@example.com** in your Pexip Infinity's configuration.

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`

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



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" -AllowAppGuestJoinsAsAuthenticated $true -AadApplicationIds "c054d1cb-7961-48e1-b004-
389e81356232"
```

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

The following commands may take up to 6 hours to come into effect.

1. Assign lobby-bypass capabilities to the **Trusted app** and specify the "Alternate VTC dialing instructions". This command takes the form:
   ```
   New-CsVideoInteropServiceProvider -Name Pexip -TenantKey "<address>" -InstructionUri "<link>" -
   AllowAppGuestJoinsAsAuthenticated $true -AadApplicationIds "<Trusted 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" -
   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
   ```

The following commands are also available when troubleshooting or maintaining your system:

- `Get-CsVideoInteropServiceProvider`: list the existing interop service providers.
- `Set-CsVideoInteropServiceProvider`: modify an interop service provider (see Changing the alternative dialing instructions for more information).
- `Remove-CsVideoInteropServiceProvider`: remove a provider.
- `Get-CsTeamsVideoInteropServicePolicy`: list the existing service policies.
- `Get-CsOnlineUser -Identity alice@example.com`: returns user information.

Next steps

You must now complete the configuration within Pexip Infinity as described in Configuring Pexip Infinity as a Microsoft Teams gateway.

Configuring Pexip Infinity as a Microsoft Teams gateway

Integrating Microsoft Teams with Pexip Infinity provides any-to-any video interoperability with Microsoft Teams.

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).

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

Prerequisites

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 Configuration > 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 your Microsoft Azure tenants

A Microsoft Azure 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 Azure tenant — your Tenant ID specifically — in Pexip Infinity and then associate your Azure 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.

To configure your Azure tenant:

1. Go to **Call Control > Microsoft Azure Tenants** and select **Add Azure 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 Azure tenant.
Azure tenant ID	The Azure 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 Azure tenant is now available to associate with your Teams Connectors.

Configuring your Teams Connector addresses

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

To configure the address of your Teams Connector:

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 IP address or FQDN of the Teams Connector to use when placing outbound calls to Microsoft Teams. This is the address you stored in **$PxTeamsConnFqdn** 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. pexip-TeamsConn-eu.teams.example.com.
Port	The IP port to connect to on the Teams Connector.
Azure tenant	Select the Azure tenant to associate with this Teams Connector.

3. Select **Save**.

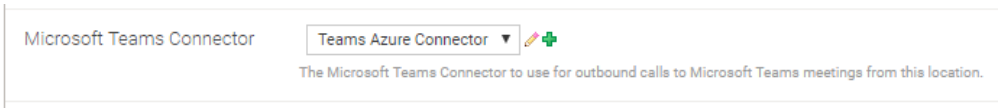
This Teams Connector is now available to associate with any system locations, Call Routing Rules or Virtual Receptions that you configure (as described below) to handle Teams conferences.

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. However, 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 Configuration > 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

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 meeting code 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 Pexip Distributed Gateway**: here you only use one or more Call Routing Rules to route incoming calls for specific alias patterns — that will typically include the meeting code — directly into the relevant Teams conferences.

The meeting code for a Teams meeting is always 9 digits.

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 meeting code which it then sends to the Teams Connector for verification. You can then optionally transform the meeting code to meet any dial plan requirements, before the Pexip Distributed Gateway then matches the (optionally transformed) meeting code and routes the caller to the appropriate Teams conference.

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).

To configure the Virtual Reception:

1. Go to **Service Configuration > 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 ***Microsoft Teams***.
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 **$PxNodeFqdns** 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 **ivr** and **d** parameters that were specified in the **InstructionUri** switch in the **New-CsVideoInteropServiceProvider** command when installing the Teams Connector, i.e. the alias of the Virtual Reception is **ivr@d**, for example **teams@example.com**.

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 **ip** parameter, e.g. 198.51.100.40.

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 Pexip Distributed 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-digit meeting code, for example the pattern could be just **<meeting code>** or **<meeting code>@<domain>**, for example **234567890@example.com** to access a Teams conference with a meeting code 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 meeting code which it can then use to connect to the conference.

Setting up the rule for direct and indirect routing

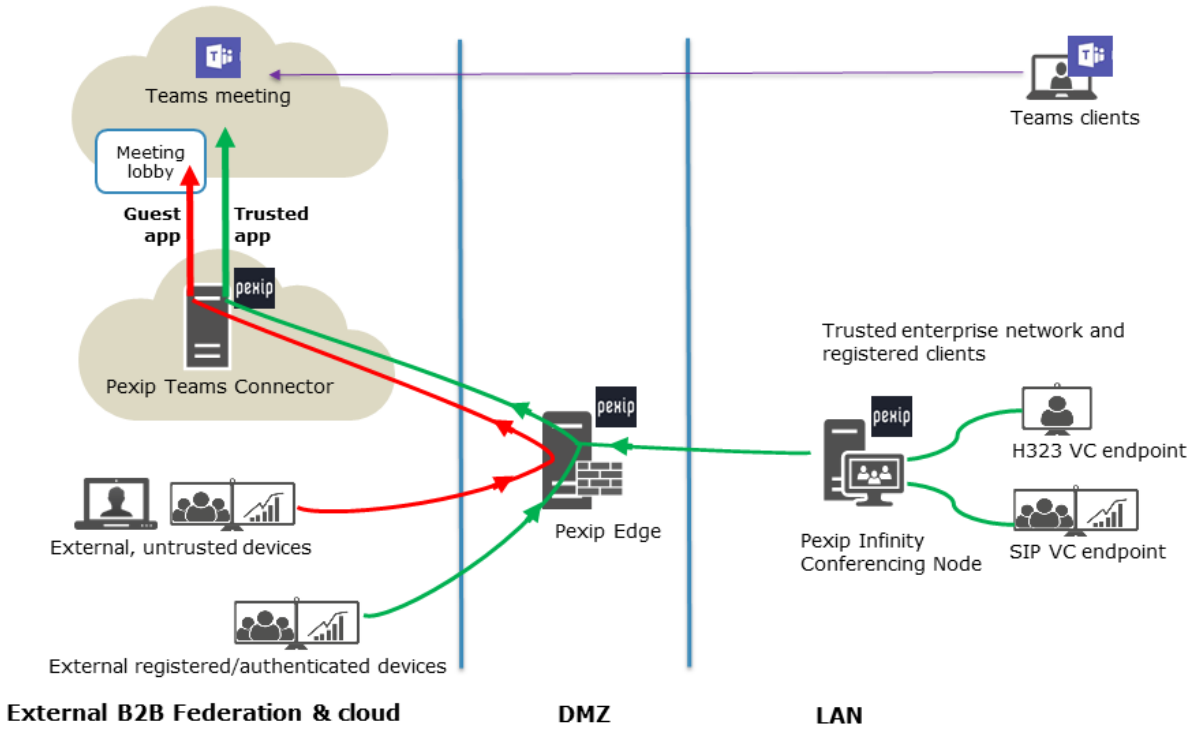
In the examples used here, the alias pattern that the rule needs to match is **<meeting code>** or **<meeting code>@<domain>** in both the indirect and direct routing scenarios. For direct routing this is the alias format that is dialed, and for indirect routing it is simply the meeting code that is 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 use the **Trusted app** to route the call into the Teams meeting (see Authorize Trusted app to bypass Teams lobby and configure dialing instructions for more information). The **Guest app** is used 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 **Service Configuration > 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.	☑	☑	☑

Option	Description	Rule #1	Rule #2	Rule #3
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 ***Any Location***.	"Internal network" location	***Any location***	***Any location***
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.	☐	☑	☐
Match Infinity Connect Match SIP Match Lync / Skype for Business (MS-SIP) / Match H.323	Select one or more of **Match Infinity Connect (WebRTC / RTMP)**, **Match SIP**, **Match Lync / Skype for Business (MS-SIP)** and **Match H.323** 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.	☐	☐	☐
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 **<meeting code>** or **<meeting code>@example.com**, you could use: **(\d{9})(@example\.com)?** (Note that Teams meeting codes are always 9 digits long. Putting **( )?** around the @domain part of the regex makes that part of the dial string optional.)	**(\d{9})(@example\.com)?**		
Destination alias regex replace string	If required, enter the regular expression string to transform the originally dialed (matched) alias into the 9-digit meeting code 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 <meeting code> element of the alias.	**\1**		
Call capability	To support incoming calls via SIP, H.323 and WebRTC/RTMP, ***Same as incoming call*** is recommended as this makes the most efficient use of Pexip Infinity resources. If you also want to support calls from Skype for Business / Lync, then you should select ***Main video + presentation*** 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 ***Main video + presentation***).	Select these options as appropriate		

Option	Description	Rule #1	Rule #2	Rule #3
Theme	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.	Select these options as appropriate		
Call target	Select ***Microsoft Teams meeting***.	***Microsoft Teams meeting***		
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 **$PxNodeFqdns** variable in the initialization script.	Select the location that contains the Conferencing Nodes that are referenced in the **$PxNodeFqdns** 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	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. 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. Trusted participants are automatically admitted into a Teams conference.	☑	☑	☐

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 meeting code of the Teams conference they want to join. The Pexip Distributed 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 meeting code of 234567890).

Dial plan conflicts

If the 9-digit Teams meeting code and **<meeting code>@<domain>** style aliases might overlap with your existing dial plan, you can use something more specific for your aliases such as **teams.<meeting code>@<domain>** i.e. "teams." followed by the meeting code 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**

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

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

  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.<meeting code>@<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 meeting 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" -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).

Interoperability and deployment features

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.

Call and participant status

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**) will show them as connected to the same external conference — which is identified as "Teams meeting <meeting code>".
- When viewing the participant status for a gateway call, the other participants in the conference are listed. The format of their alias indicates the type of participant:
  - user:<id> represents another Teams client 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.

  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.
  - 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 3 video streams at different resolutions to the Teams Connector, as requested by Teams.
  - You also see a presentation stream if any participant is sharing content.

Media streams

Type	Start time	Tx codec	Tx bitrate (kbps)	Tx resolution	Tx packets sent	Tx packets lost	Tx current packet loss	Tx jitter (ms)	Rx codec	Rx bitrate (kbps)	Rx resolution	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	H.264 Baseline (mode 0)	728	640x360	248824	0	0.0%	0.2
Video	2018-11-29 16:00:31 (GMT)	Off	0		0	0	0.0%	0.0	H.264 Baseline (mode 0)	2216	1280x720	305836	0	0.0%	0.2
Video	2018-11-29 16:00:31 (GMT)	Off	0		0	0	0.0%	0.0	H.264 Baseline (mode 0)	792	640x360	24881	0	0.0%	0.4
Video	2018-11-29 16:00:31 (GMT)	H.264	787	640x360	686067	11	0.0%	1.3	Off	0		0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	H.264	2479	1280x720	31762	0	0.0%	2.2	Off	0		0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0	0	0.0%	0.0	Off stage	0		0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0	0	0.0%	0.0	Off stage	0		0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0	0	0.0%	0.0	Off stage	0		0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0	0	0.0%	0.0	Off stage	0		0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0	0	0.0%	0.0	Off stage	0		0	0	0.0%	0.0
Video	2018-11-29 16:00:31 (GMT)	Off	0		0	0	0.0%	0.0	Off stage	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 control (e.g. disconnect, mute or transfer) any of the other participants connected to the Teams conference.

Additional 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 Pexip Distributed Gateway calls). Any external participants who are connected directly to the Teams conference do not consume a license.
- Content sharing is supported between Teams clients (via VbSS) and gatewayed participants.
- You cannot limit the **Maximum outbound call bandwidth** (the call leg towards the Teams Connector) — it is managed automatically by Pexip Infinity.

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 Teams Connector's certificate
- Changing the pool name or node FQDNs of the associated Conferencing Nodes (the name referenced by the **$PxNodeFqdns** variable in the initialization script)

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, or if you need to modify the number of Teams Connector instances.

Upgrading the Teams Connector to the latest software

If you need to upgrade the Teams Connector, you must redeploy it using your original configuration scripts (with any configuration changes if required), and with the latest application software files, as described below.

If you have deployed multiple Teams Connectors, follow the same redeploy process for each Teams Connector.

Redeploying the Teams Connector

The Teams Connector redeployment process described below is required when you want to:

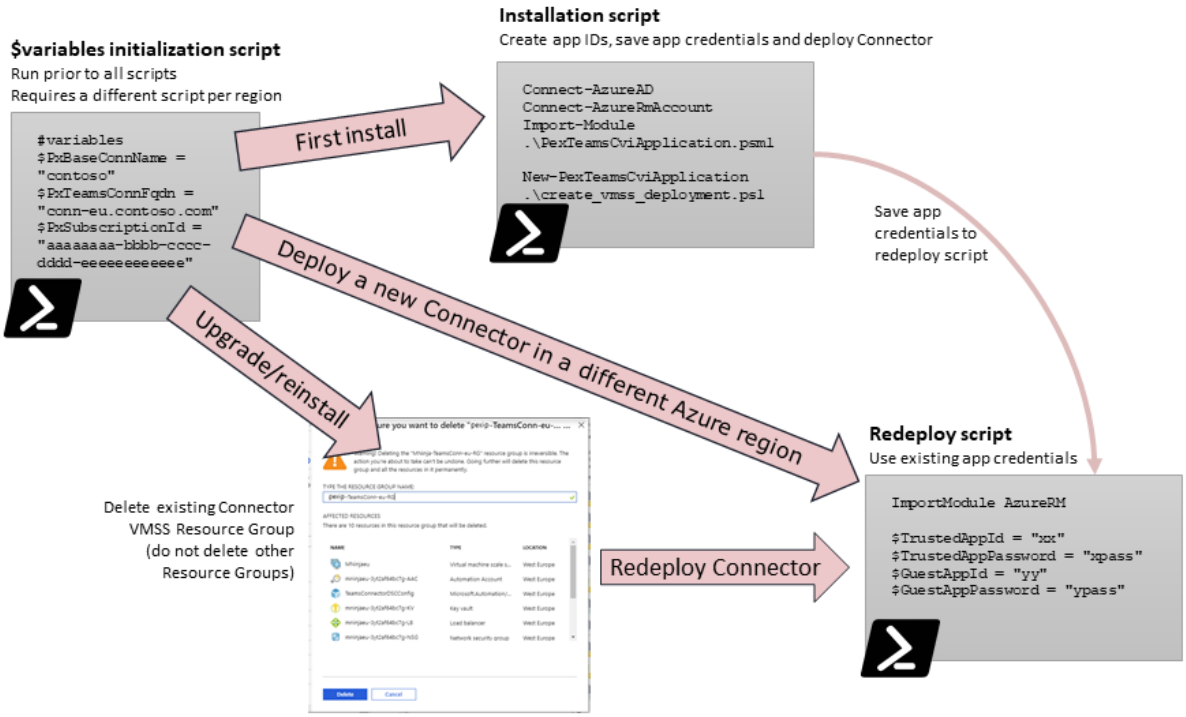
- upgrade the Teams Connector software
- change the Teams Connector's configuration (such as its certificate, or the names of the associated Conferencing Nodes)
- deploy a new Teams Connector in a different Azure region.

Note that there is 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.

There are three main steps to redeploying the Teams Connector:

1. Check that you have the latest relevant version of the Teams Connector software, and your original initialization and redeploy scripts.
2. Remove **some** of the existing Teams Connector resources from Azure.
3. Reinstall the Teams Connector software.

These steps are shown in the following diagram and are then described in detail below.



Check Teams Connector software and retrieve your original scripts

1. Download the latest relevant version of the **Teams Connector ZIP** file (Pexip_Infinity_Connector_For_Microsoft_Teams_v20_<build>.zip) from the Pexip support site.
   Ensure that the Teams Connector version you download is the same version as your Pexip Infinity deployment.
2. Extract the files to a folder 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 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:
   - Compare your version of the redeploy script to the one that is shown below, and verify that your redeploy script was updated with the App IDs and passwords for your deployment (to ensure that upgrades/redeployments can be done using the same App IDs):
     - When the installation script ran, it generated some output that listed the App IDs and credentials, similar to this:

       ```
       ### App ID and credentials MUST be saved in the secondary script ###"

       $TrustedAppId = "c054d1cb-7961-48e1-b004-389xxxx32"
       $TrustedAppPassword = "bXl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3JkbXl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3JkbXl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3Jk=="
       $GuestAppId = "18c0476d-ee99-4673-a6da-xxxxx"
       $GuestAppPassword = "eHl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3JkbXl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3JkbXl2ZXJ5bG9uZ3NlY3JldHBhc3N3b3Jk=="
       ```

     - These output lines should have been used to replace the four existing lines in the redeploy script that say:

       ```
       $TrustedAppId = ""
       $TrustedAppPassword = ""
       ```

```
$GuestAppId = ""
$GuestAppPassword = ""
```

This means that when you run the redeploy script, you will not rerun the commands that create the apps. Instead you will set the variables to the IDs and passwords of the apps that you created the first time.

- You should use this 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 this 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 some Teams Connector resources from Azure

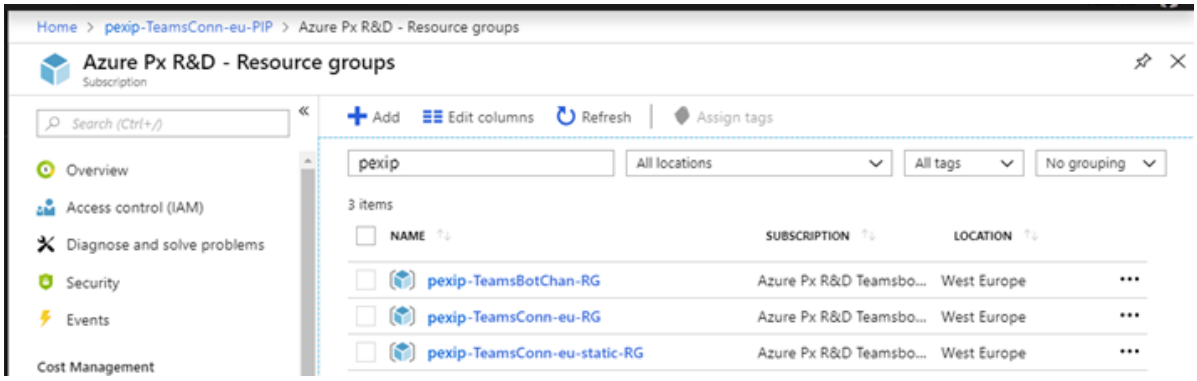
Before redeploying or upgrading a Teams Connector, you must remove **some** of the existing Teams Connector resources from Azure:

1. 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>-TeamsBotChan-RG**: this contains the Bot channel registrations and must **not** be deleted.
   - **<prefix>-TeamsConn-<region>-RG**: this contains the Teams Connector instances (Virtual Machine scale set) and is to be deleted in the next step.
   - **<prefix>-TeamsConn-<region>-static-RG**: this contains the public IP address of the Teams Connector and must **not** be deleted.

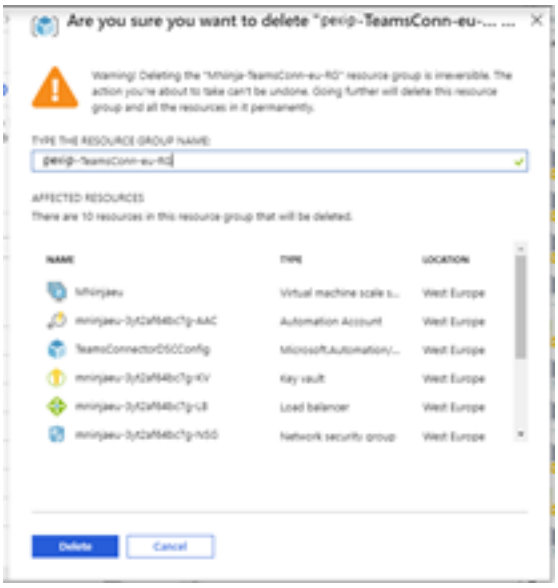
   If you have deployed a Teams Connector in multiple regions you will see several **<prefix>-TeamsConn-...** resource groups.



2. Select the **<prefix>-TeamsConn-<region>-RG** resource group.

   (Do **not** select any resource groups with **TeamsBotChan** or **Static** in its name.)

   If you have many Teams Connectors in multiple regions, select the resource group with the **<region>** name you want to delete.
3. Select **Delete resource group** and confirm with the name of the resource group.



Your Teams Connector will stop working when you confirm this step.

4. The deletion may take a few minutes to complete. When the resource group is deleted you can start to redeploy the Teams Connector as described below.

Redeploy the Teams Connector

To redeploy the Teams Connector:

1. 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.
2. 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.

```
# Connect to PowerShell AzureRm
# Set execution policy for the current PowerShell process
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process

# Connect to Azure Resource Manager PowerShell (in same window to reuse variables)
Import-Module AzureRM -MinimumVersion 6.0.0
Connect-AzureRmAccount

# Azure RM commands
# Before running the following commands, update the following 4 lines/variables with the App IDs and passwords
# that were output when the installation script was run
$TrustedAppId = ""
$TrustedAppPassword = ""
$GuestAppId = ""
$GuestAppPassword = ""

# Change context to the Pexip Subscription and set the Trust/Guest credentials
Set-AzureRmContext -SubscriptionId $PxSubscriptionId

$TrustedAppSecurePassword = ConvertTo-SecureString -AsPlainText $TrustedAppPassword -Force
```

```
$TrustedAppCred = New-Object System.Management.Automation.PSCredential -ArgumentList
$TrustedAppId,$TrustedAppSecurePassword

$GuestAppSecurePassword = ConvertTo-SecureString -AsPlainText $GuestAppPassword -Force
$GuestAppCred = New-Object System.Management.Automation.PSCredential -ArgumentList
$GuestAppId,$GuestAppSecurePassword

# 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 variables above) for the Windows VM
$PxWinAdminSecurePassword = ConvertTo-SecureString -AsPlainText $PxWinAdminPassword -
Force
$PxWinAdminCred = New-Object System.Management.Automation.PSCredential -ArgumentList
$PxWinAdminUser,$PxWinAdminSecurePassword

# Optionally if you do not prefer to have a password set as a variable, use Get-
Credential
# $PxWinAdminCred = Get-Credential

# Create Resource group for Teams Connector Load Balancer (per region)
# This stores the public IP address of the Teams Connector Load Balancer
# so that the address can be re-used on upgrade or redeploy
New-AzureRmResourceGroup -Location $PxAzureLocation -ResourceGroupName
$PxTeamsConnStaticResourceGroupName -Force

# Create Resource group for Teams Connector VMSS (per region)
# This section of the script is region specific, ensure that the initial variables
# are set with the correct values for $PxAzureLocation, $PxTeamsConnFqdn and
$PxVmssRegion
New-AzureRmResourceGroup -Location $PxAzureLocation -ResourceGroupName
$PxTeamsConnResourceGroupName

# Ensure required version of the Carbon Powershell module is used
(Get-Content TeamsConnectorDSCConfig.ps1).replace('Install-Module -Name Carbon -
AllowClobber', 'Install-Module -Name Carbon -RequiredVersion 2.6.0 -AllowClobber') | Set-
Content TeamsConnectorDSCConfig.ps1

# 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 -
TrustedAppCredential $TrustedAppCred -GuestAppCredential $GuestAppCred -
StaticResourcesResourceGroupName $PxTeamsConnStaticResourceGroupName -
PublicIPAddressResourceName "$($PxBaseConnName)-TeamsConn-$($PxVmssRegion)-PIP" -
IncidentReporting $PxTeamsConnIncidentReporting -Encryption $PxTeamsConnDiskEncryption -
RdpSourceAddressPrefixes $PxMgmtSrcAddrPrefixes -PexipSourceAddressPrefixes
$PxNodesSourceAddressPrefixes

# 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 AzureAD and
AzureRM)
#
# Setting subscription
```

```
Set-AzureRmContext -SubscriptionId $PxSubscriptionId

# Getting Network Security Group Resource ID
$nsgResId = (Get-AzureRmResource -ResourceGroupName $PxTeamsConnResourceGroupName -
ResourceType Microsoft.Network/networkSecurityGroups).ResourceId

# Getting Public IP details
$publicIpAddress = (Get-AzureRmPublicIpAddress -ResourceGroupName
$PxTeamsConnStaticResourceGroupName).IpAddress
$publicIpFqdn = (Get-AzureRmPublicIpAddress -ResourceGroupName
$PxTeamsConnStaticResourceGroupName).DnsSettings.Fqdn

# 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 CNAME from
your official hostname"

Write-Host
Write-Host "1) Setup a DNS CNAME for $($PxTeamsConnFqdn) pointing to "
Write-Host "   $($publicIpFqdn)"
Write-Host
Write-Host "   When this is done, and you can confirm a DNS lookup of $($PxTeamsConnFqdn)
resolves to"
Write-Host "   your Public IP of the load balancer ($($publicIpAddress)) - you are ready
to proceed."

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

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 / Lync 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

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" -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"`

and the command to apply the revised **InstructionUri** parameter would be:

Note that when using `Set-CsVideoInteropServiceProvider`, the first parameter is called `-Identity`, not `-Name`.

```
Set-CsVideoInteropServiceProvider -Identity Pexip -InstructionUri
"https://px.vc.example.com/teams/?conf={ConfId}&ivr=teams&d=example.com&ip=198.51.100.40&w"
```

This change may take up to 6 hours to come into effect.

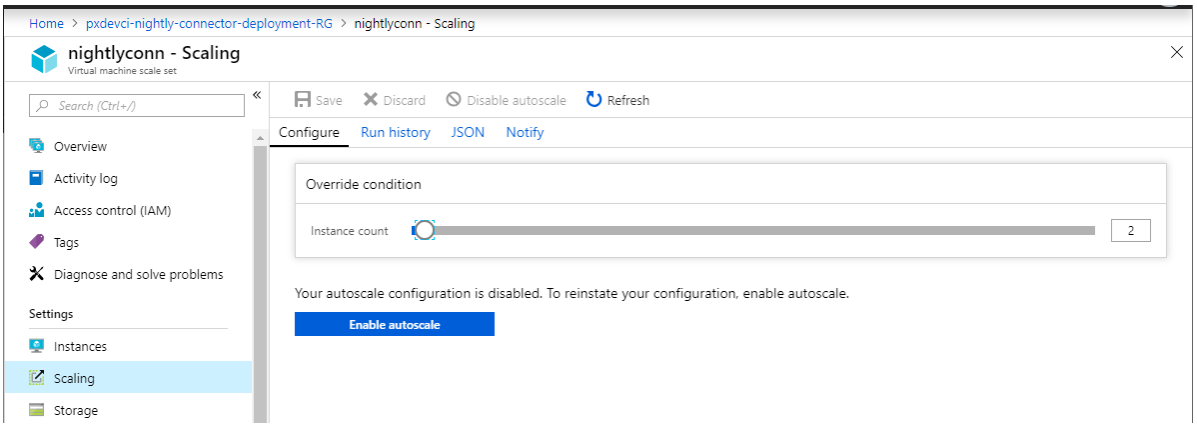
Changing the call capacity of a Teams Connector

When you install a Teams Connector application in Azure, you initially specify the number of Teams Connector instances (VMs) to deploy.

You can subsequently modify the number instances via the Azure portal, to reflect changing capacity requirements.

To change the 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**.
3. Use the slider in the Configure tab 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.



Adding extra Teams Connectors in other 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-prem 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.

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.
3. Run the new initialization script for your new Azure region.
4. Run the redeploy script that you produced after the initial installation. This script should already have the variables for the trusted and guest App IDs and passwords 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.
5. Update DNS with the new Teams Connector's name and IP address.

Note that you do **not** need to create any new apps 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 Configuration > 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.

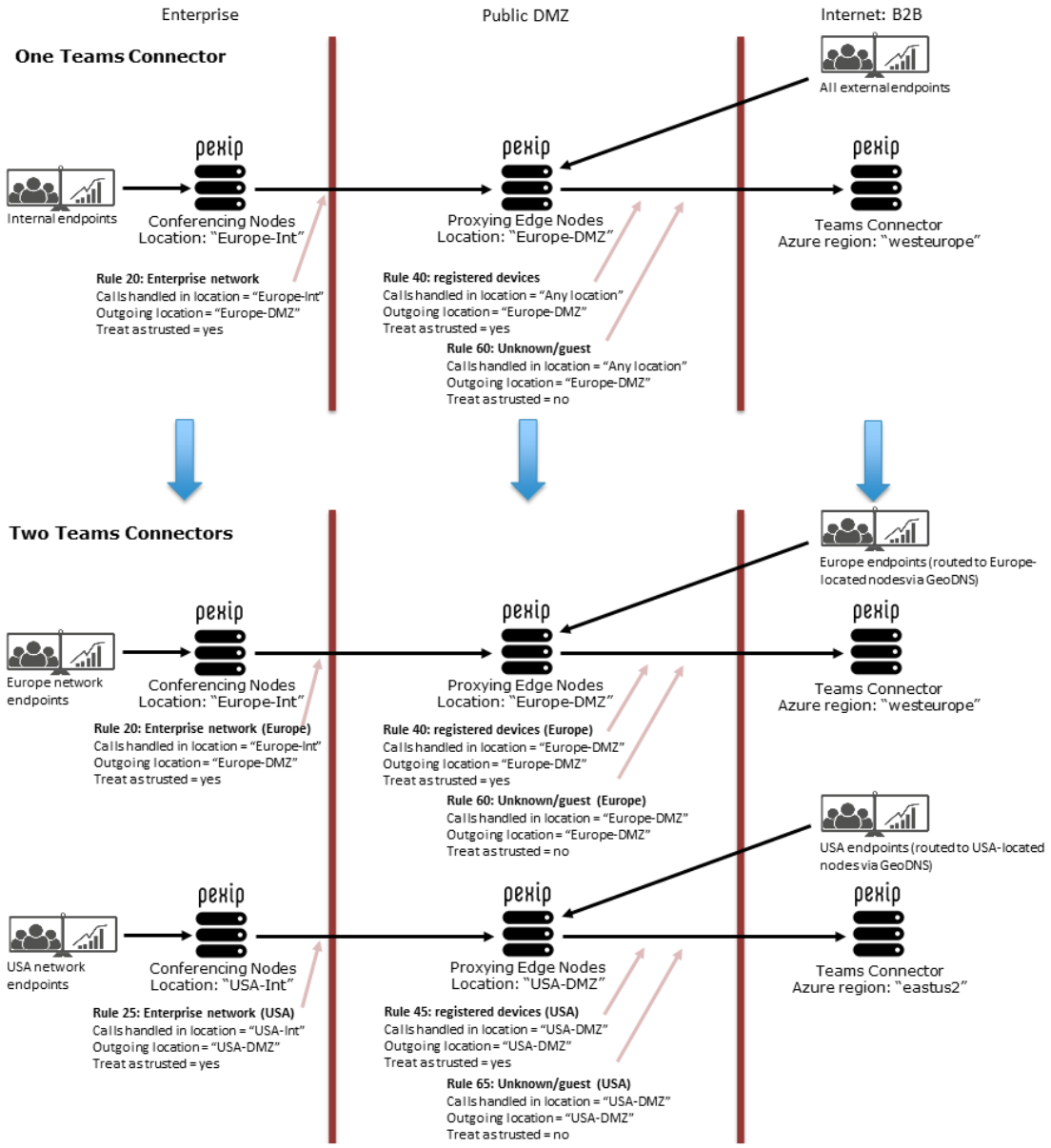
   For example, if you previously only had a single location (called "Europe-DMZ") containing all of your Conferencing Nodes and that location was also configured as the **Outgoing location** on all of your rules, then all calls would have been received in — and routed out of — those nodes to your single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



Note that it is not necessary to modify existing or create new Virtual Receptions — there is minimal performance overhead by always using a single location and Teams Connector to resolve Teams meeting codes entered into a Virtual Reception.

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.

You can change or enable 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. Set **Action** to **Allow** or **Deny** as appropriate.

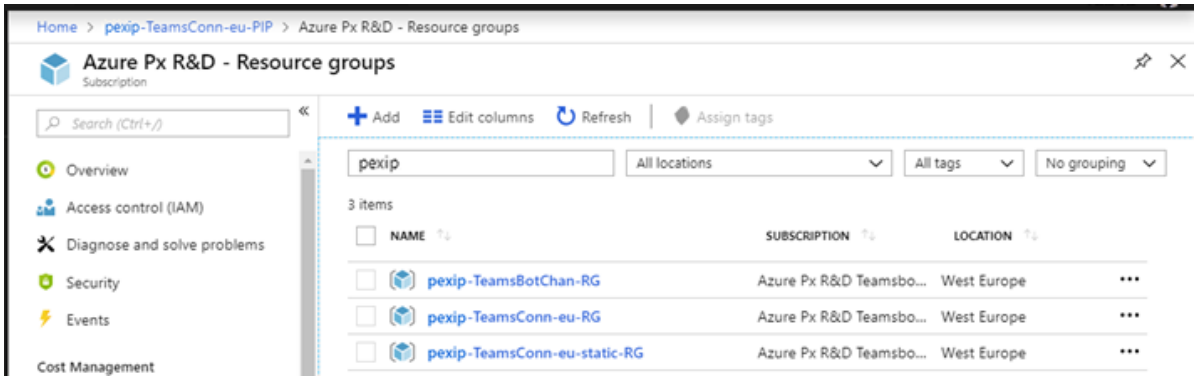
5. Select **Save**.
6. Update the **$PxMgmtSrcAddrPrefixes** installation variable in your stored initialization script to match your changes applied to the NSG to keep the script in sync with your actual deployment.

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.

1. Delete all of the resource groups from Azure:
   a. 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>-TeamsBotChan-RG**: this contains the Bot channel registrations.
   - **<prefix>-TeamsConn-<region>-RG**: this contains the Teams Connector instances (Virtual Machine scale set).
   - **<prefix>-TeamsConn-<region>-static-RG**: this contains 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.
2. Remove the Trusted and Guest App registrations from Azure:
   a. Within the Azure portal, select **Azure Active Directory**.
   b. Under **Manage**, select **App Registrations**.
   c. Select the Trusted app registration with the prefix of the Teams Connector you want to delete, and then **Delete** it.
   d. Select the Guest app registration with the prefix of the Teams Connector you want to delete, and then **Delete** it.
3. Remove the Cloud Video Interop service provider from your tenant:
   a. Start a PowerShell session and run the following commands, where **<tenant_name>** needs to be your onmicrosoft.com domain for your tenant:
   ```
   Import-Module SkypeOnlineConnector
   $sfbSession = New-CsOnlineSession -OverrideAdminDomain "<tenant_
   name>.onmicrosoft.com"
   Import-PSSession $sfbSession
   ```
   b. Run the command:
   ```
   Remove-CsVideoInteropServiceProvider -Identity Pexip
   ```
4. Remove references to the Teams Connector from the Pexip Management Node:
   a. Change or delete any locations, Virtual Receptions or Call Routing Rules that are using the Teams Connector.
   b. Go to **Call Control > Microsoft Teams Connectors** and delete the Teams Connector address.

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.

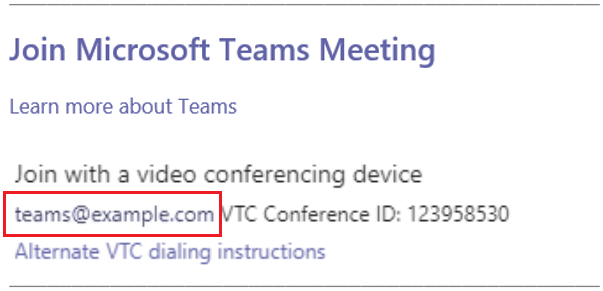
Example deployment scenario

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, H.323 and Infinity Connect clients 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**.

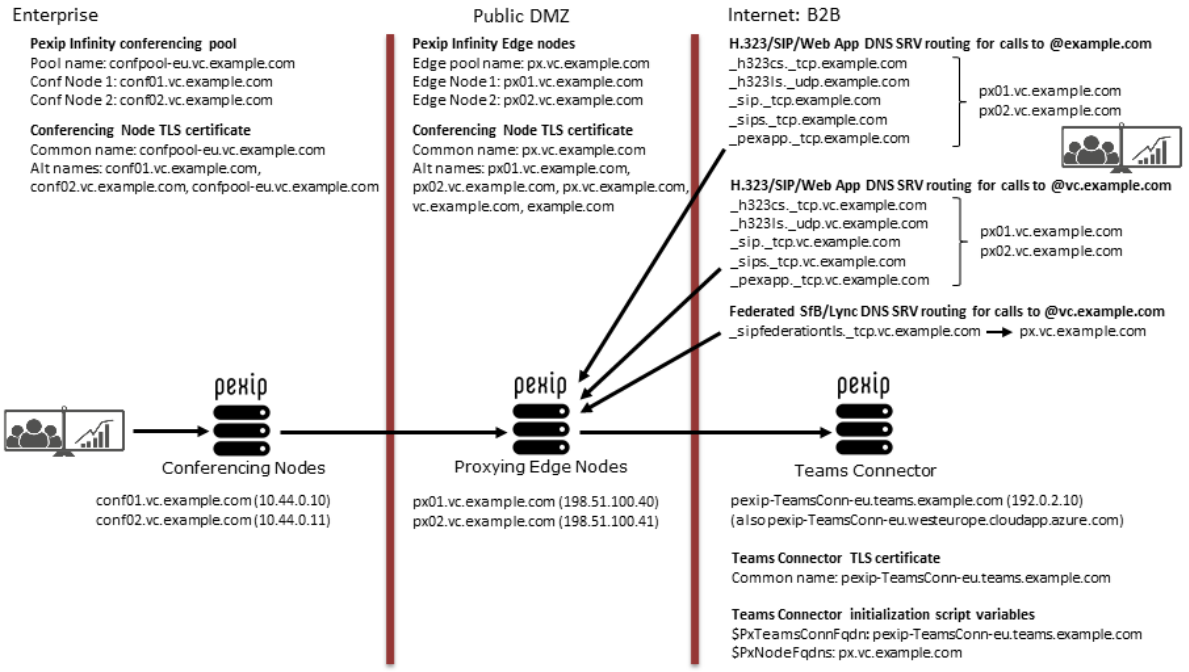
  Your choice of routing via your primary domain (**example.com** in this case) and/or via your Pexip subdomain (**vc.example.com**) depends on your existing environment and dial plan requirements. For example, if your environment already includes DNS records and routing for SIP and H.323 devices to your 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, H.323 and Infinity Connect clients 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.



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.
- 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).

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.

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 is what you will use as the value of **$PxTeamsConnFqdn** in the variables initialization script.
   - The certificate's subject name must match 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 a different domain to your Teams and Pexip Infinity deployments.
   - 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.
   - It can be a wildcard certificate.
2. Request a certificate for that name and generate the certificate in PFX format. Any intermediate certificates must also be in the PFX file.

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 Infinity Connect clients (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 5060 conf01.vc.example.com.
_pexapp._tcp.example.com. 86400 IN SRV 10 10 5060 conf02.vc.example.com.
```

In addition, those VTC devices and Infinity Connect clients 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 5060 conf01.vc.example.com.
_pexapp._tcp.vc.example.com. 86400 IN SRV 10 10 5060 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 Infinity Connect clients (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.)

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 Infinity Connect clients 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 / Lync 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 meeting 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.

Troubleshooting Microsoft Teams and Pexip Infinity integrations

This topic describes any limitations and provides troubleshooting guidance when integrating Microsoft Teams with Pexip Infinity.

Installation issues

This section describes issues that might occur when running the Teams Connector installation script.

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 bot channel registration 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.

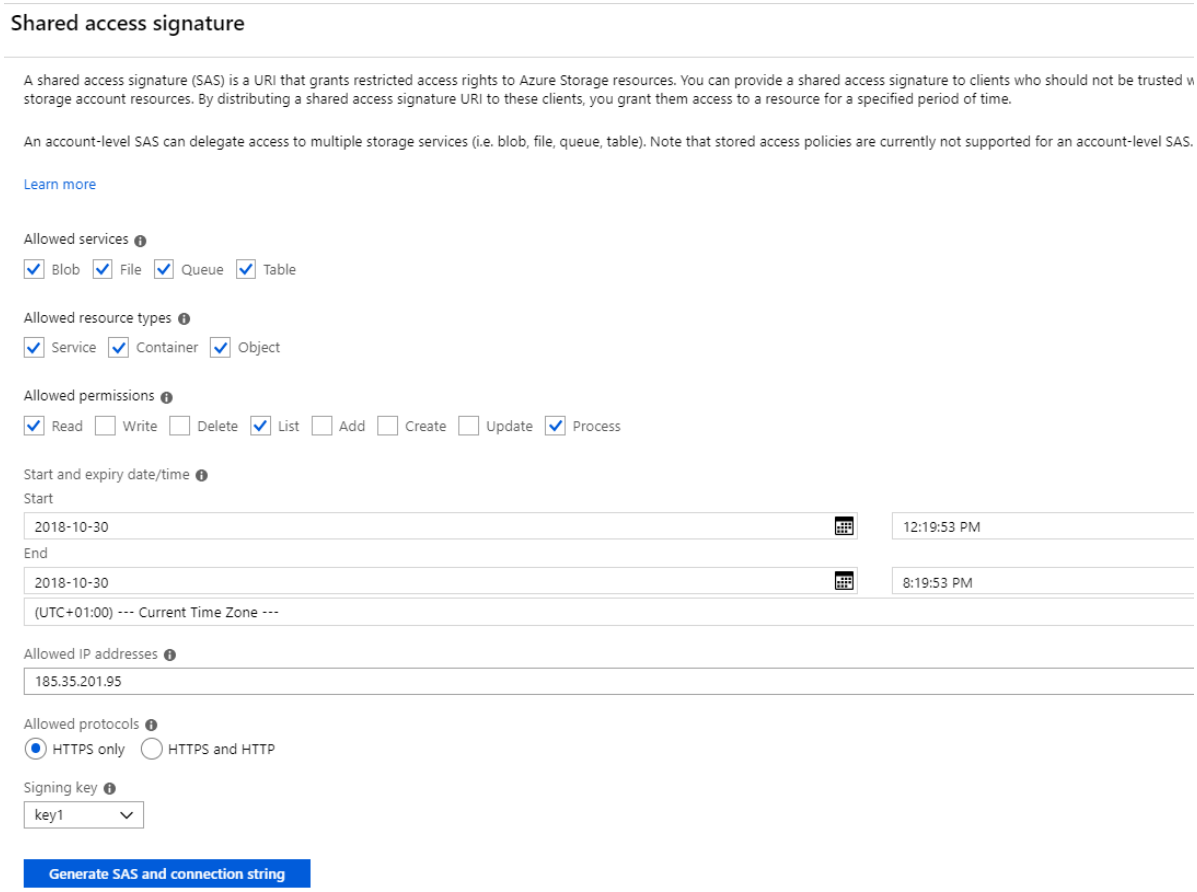
Obtaining Teams Connector logs

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 Management Node.

Note that logs and incident reports are kept for 30 days on the Teams Connector.

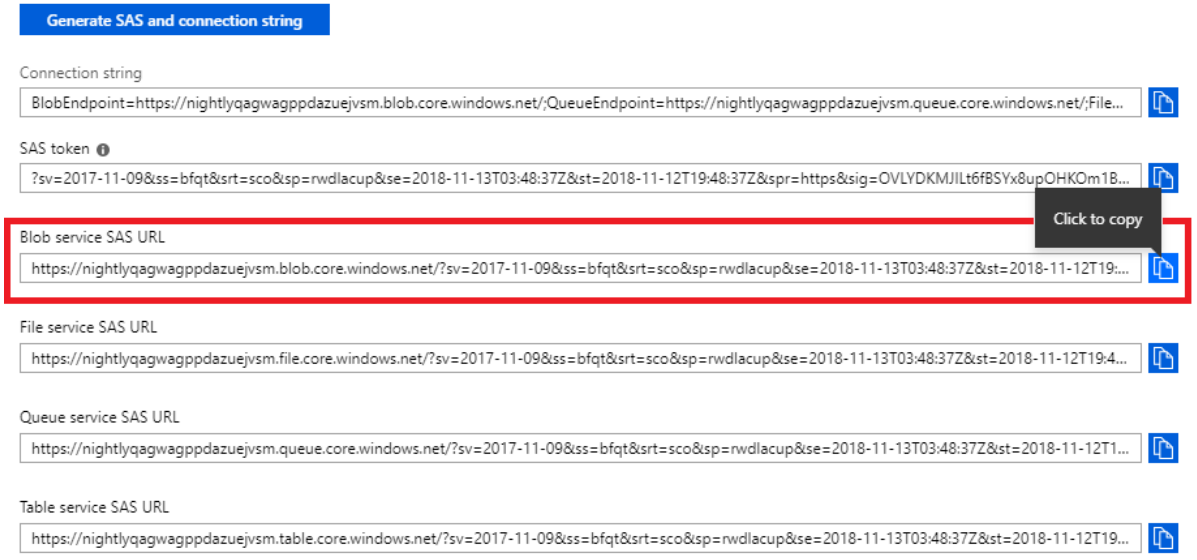
To get the logs and snapshot:

1. Get the Teams Connector logs:
   a. In the Azure portal, for your subscription, select the Teams Connector resource group that contains your Virtual machine scale set, Load balancer, Network security group and so on.
   b. Select the **Storage account** in that resource group.
      Note that the logs themselves are stored as Blobs in a Logs container within the storage account (typically there are more logs folders than actual VM instances). However, you do not need to access these individual logs as you will generate a URI that grants restricted access to those logs.
   c. In **Settings**, select **Shared access signature**.
   d. In the **Allowed permissions**, clear (untick) the following boxes: ***Write***, ***Delete***, ***Add***, ***Create***, ***Update***.
   e. Set the **Allowed IP addresses** to 185.35.201.95 — this is the address of the Pexportal support site.
   f. Leave everything else selected according to the default settings unless requested otherwise.
   g. Select **Generate SAS and connection string**.



A list of strings and URLs are generated.

h. Copy the **Blob service SAS URL**.



2. Get a snapshot from the Pexip 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 just the most recent data:

i. Type in, or adjust the slider to the minimum number of hours of diagnostic data to be downloaded. The number of hours available will vary depending on the logs available on the system.
ii. Select **Download limited duration snapshot**.
iii. 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. Select **Upload Files**.
   c. Fill in the page details:

Ticket number	Enter your support ticket number.
Azure SAS URL	Paste in here the **Blob service SAS URL** from the Azure portal and select **Fetch dates**.
Select a file to upload	Select **Choose file** and select the diagnostic snapshot file you downloaded from the Pexip Management Node.

   d. Select **Upload**.
   You see a message that the snapshot file has been uploaded and the Azure logs are queued for download.
   You can now exit Pexportal.

Call failures (invalid conference ID and rejected calls)

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 digit VTC 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.

Symptom	Possible cause	Resolution
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 Install Teams Connector and Conferencing Node certificates 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 **$PxNodeFqdns** 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.	Check the Azure tenant ID configured in Pexip Infinity (**Call Control > Microsoft Azure 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 apps.

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 digit VTC 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-digit meeting code.
	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 Azure 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 apps.
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 Install Teams Connector and Conferencing Node certificates 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 **$PxNodeFqdns** 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.

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

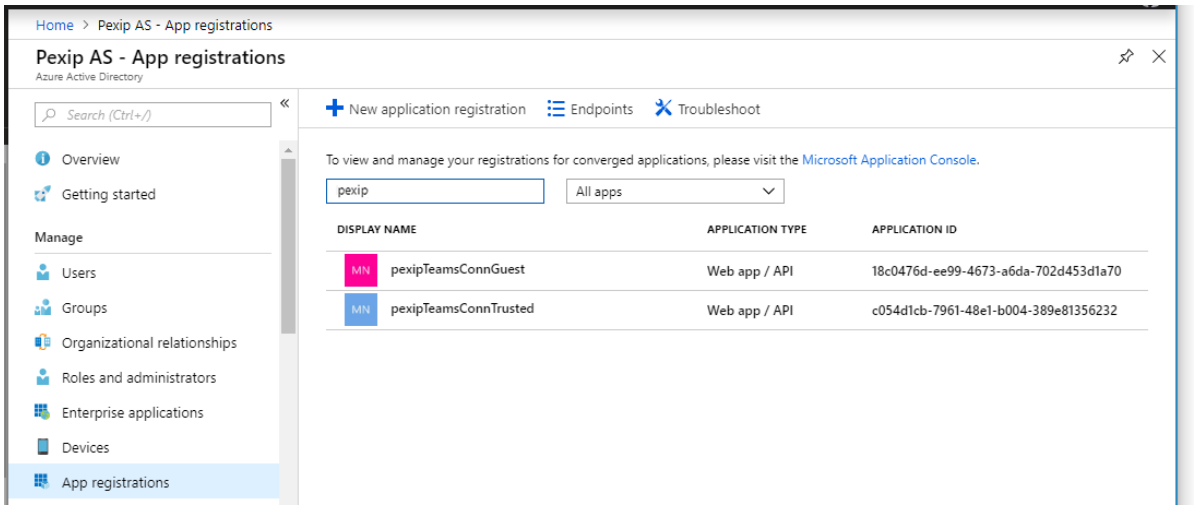
Your Pexip CVI applications need to be granted permission to enable access to Microsoft Teams in an Office 365 tenant, as described in Authorize Pexip CVI applications to join Teams meetings.

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:

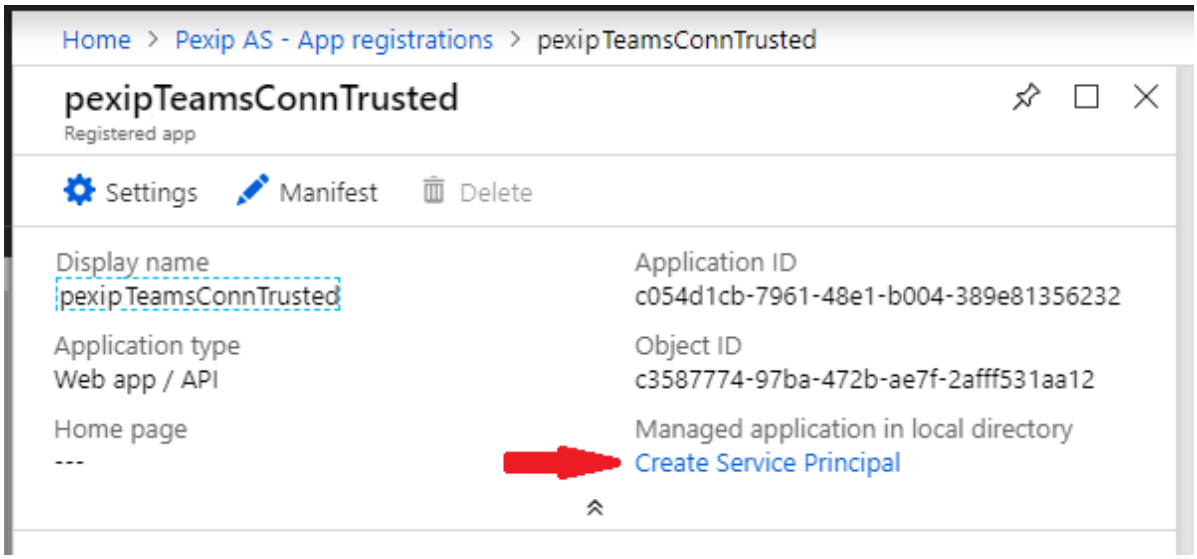
1. Go to the Azure portal (https://portal.azure.com) and sign in.
2. Go to **Azure Active Directory > App Registrations**.
3. Search for the prefix you used when deploying your Teams Connector, and it will list your app registrations as shown below (your app names will be different).



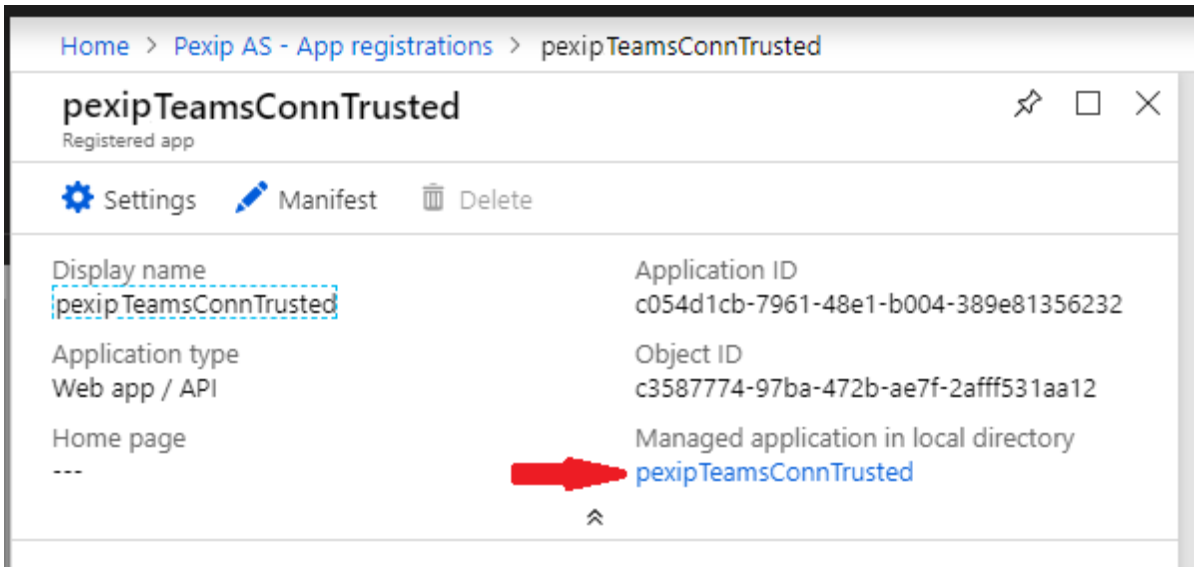
Note that if you have deployed the Teams Connector several times, old app registrations will also show — even if you have deleted the **botchannelregistration** resource group. If required, you can delete these old registrations.

4. To check if consent has been granted in your Tenant, select each registration from the list.

   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 **Bot Channel Registration**, however in most enterprises it will be in the same Azure AD.)



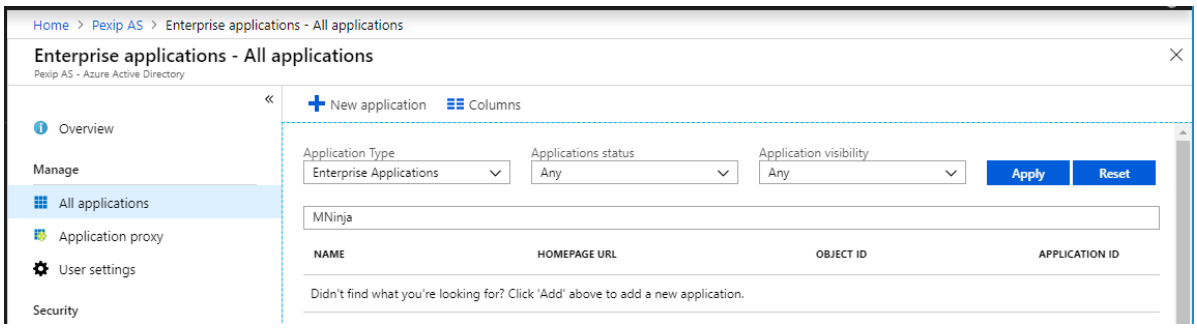
If consent has been granted in your tenant, the registration will show a reference to the enterprise app ("pexipTeamsConnTrusted" in the example below).



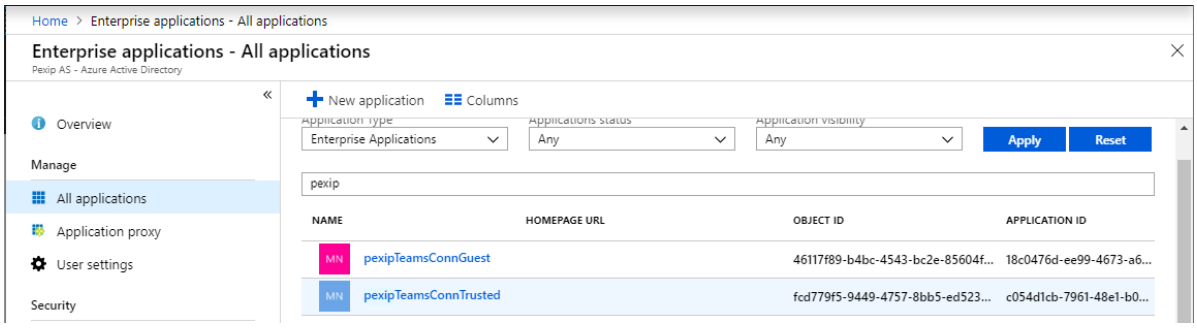
View authorized enterprise apps

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 apps have had consent granted, you will see a list of apps as shown below (your app names will be different):



3. If your Teams Connector apps are not shown, you must provide consent to them as described in Authorize Pexip CVI applications to join Teams meetings.
   Your apps will then appear in the list of enterprise applications as shown in the second example screenshot above.
   Also when you view the apps 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.