

Azure Active directory Deployment plan

**Application Proxy**

Abstract

This plan covers the deployment requirements, guidance, and plans necessary to deploy <Application Name> to <Insert Company Name> using Application Proxy

MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, our provision of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The descriptions of other companies’ products in this document, if any, are provided only as a convenience to you. Any such references should not be considered an endorsement or support by Microsoft. Microsoft cannot guarantee their accuracy, and the products may change over time. Also, the descriptions are intended as brief highlights to aid understanding, rather than as thorough coverage. For authoritative descriptions of these products, please consult their respective manufacturers.

© 2015 Microsoft Corporation. All rights reserved. Any use or distribution of these materials without express authorization of Microsoft Corp. is strictly prohibited.

Microsoft, Active Directory Domain Services, Azure Active Directory, Azure AD Premium, Office, Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Table of Contents

[1 Overview 1](#_Toc508930344)

[2 Assessment 1](#_Toc508930345)

[2.1 Deployment Preparation Activities 1](#_Toc508930346)

[2.2 Organizational Governance and Security Requirements 2](#_Toc508930347)

[3 Azure AD Application Proxy Solution 3](#_Toc508930348)

[3.1 Core Dependencies 4](#_Toc508930349)

[3.2 Applications Discovery 6](#_Toc508930350)

[3.2.1 Application specific details 7](#_Toc508930351)

[3.3 Pilot best practices recommendations 8](#_Toc508930352)

[3.4 Production best practices recommendations 10](#_Toc508930358)

[4 Azure AD Application Proxy Deployment Plans 11](#_Toc508930365)

[4.1 Prerequisites 12](#_Toc508930366)

[4.2 Use-case/scenario/capability requirements 12](#_Toc508930367)

[4.3 Working with existing on-premises proxy servers 13](#_Toc508930368)

[4.4 Enable Application Proxy and Deploy Connectors 13](#_Toc508930369)

[4.5 Create Connector Groups 15](#_Toc508930370)

[4.6 Publish Applications 16](#_Toc508930372)

[Choosing the right Applications type to publish 16](#_Toc508930373)

[Using Custom Domain Names 17](#_Toc508930374)

[Assign applications to your connector groups 17](#_Toc508930375)

[Assigning Users to your Application 17](#_Toc508930376)

[4.7 Functional Testing of Applications 18](#_Toc508930377)

[4.8 Enable Pre-Authentication and Single Sign On 19](#_Toc508930378)

[4.9 Working with “Other” Applications 19](#_Toc508930379)

[4.10 Enforcing Conditional Access 19](#_Toc508930380)

1. Overview

The purpose of this document is to detail the aspects of the Microsoft Azure Active Directory Application Proxy implementation as it relates to the <Update Customer Name in Doc Properties> - <PROJECTNAME> project. This document is for use by the <PROJECTNAME> Project Manager, <DELIVERYORG> Technical Specialists and <Update Customer Name in Doc Properties> Industry Solutions and IT teams.

Azure Active Directory (AD) Application Proxy helps you support remote workers by publishing on-premises applications to be accessed over the internet. Through the Azure portal, you can publish applications that are running on your local network and provide secure remote access from outside your network.

This document walks you through the steps to publish an on-premises app with Application Proxy. After you complete this article, you'll be ready to configure the application with single sign-on, personalized information, or security requirements.

1. Assessment

The discovery phase of this initiative is critical to gathering all the necessary preliminary information that will be used to prepare Azure Application Proxy to publish your internal web applications.

Having more information you can obtain on all of the applications in scope for being published, the quicker you can make them securely available, from any location.

This process should predominantly focus on both ends of the infrastructure, being your Microsoft Azure Active Directory and your Microsoft Active Directory Domain Services (AD DS) to identify applications which are in scope for Azure Active Directory Application Proxy.

The following sections are designed to cover all basis of taking Azure AD Application Proxy published applications from pilot, through to production, so aims to

The assessment workshop may also highlight common issues that might pose a problem during the configuration phase of Azure Active Directory Application Proxy. To accomplish this assessment, attend interview workshops to learn about your specific configuration. The following sections detail the results of the assessment workshop discussions.

The intent of this section is to provide with detailed information about necessary remediation activities that must be performed prior to implementing and configuring Azure Active Directory Application Proxy.

* 1. Deployment Preparation Activities

**Instructions:** the delivery of this project assumes that the following components are in place (enabled) and are functioning accordingly. Please update the following table with specific information about your customer.

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Decisions / Comments | Status | Mitigation |
| Customer has procured Azure Active Directory Premium | Azure Active Directory Premium is required for multi-factor authentication capabilities. This must be in place prior to deployment. | <Yes/No> | Enter in mitigation as applicable |
| Azure Onboarding | Directory synchronization is in place and is functional. | <Yes/No> | Enter in mitigation as applicable |

Table 1: Deployment Preparation Activities

* 1. Organizational Governance and Security Requirements

The following table notes items identified during the Assess phase of this project. They are not technology specific but require mitigation prior to implementing and configuring Azure Active Directory Application Proxy.

**Instruction:** the purpose of this section is to list the non-technical related organizational issues that might have come up during the Assess phase. If you did not identify any, you may delete this section. Otherwise, please use the following example for how you list the items found and the appropriate mitigation activities associated with it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Issue | Contributing Team/Contact | Owner | Mitigation Steps | Status |
| Corporate policy prohibits the use of a cloud identity solution for authentication | Security Office | CISO | Setup a meeting between CISO, Executive Sponsor, and other leadership contacts to discuss this in more detail. | Open |
| Azure Active Directory Pre-Authentication required | Security Office | CISO | Setup a meeting between GTP PM, Technical stakeholder and CISO to walk through the options available. | Open |
| Corporate policy prohibits corp traffic from routing directly out to the WWW through the default gateway, so must route via an outbound proxy | Security Office | CISO | Setup a meeting between GTP PM, Technical stakeholder and CISO to walk through the options available. | Open |
| Azure Active Directory Application Proxy connector host requires FIPS hardening | Security Office | CISO | Setup a meeting between GTP PM, Technical stakeholder and CISO to walk through the options available. | Open |

Table 2: Organizational Governance and Security Requirements

1. Azure AD Application Proxy Solution

Azure Active Directory Application Proxy provides the following key benefits:

**Simple**

* Rapid Deployment - By far easier to set up and secure than traditional on-premises solutions, as there’s no requirement to front end the proxy with a, perimeter firewall.
* You typically don't need to change or update your applications to work with Application Proxy. Real-time link translation can reduce time to deploy and potential impact of having to change the configuration of applications that are already ion production
* Users get a consistent authentication experience, regardless of their location. They can use the MyApps portal to get single sign-on to both SaaS apps in the cloud and your apps on-premises.

**Secure**

* Allowing inbound traffic through your perimeter firewall is no longer a requirement.
* Publishing your apps through Azure AD Application Proxy allows you to take full advantage of Azure’s rich authorization controls and security analytics.
* You get cloud-scale security and Azure security features like conditional access and multi-step, adaptive, verification.

**Improved User Experience**

* Single sign-on gives your end users the ease and simplicity of access to all the apps they need to be productive with either a single set of credentials, or no passwords, whilst providing a consistent authentication experience.

**Reduced Cost of Ownership**

* Cloud centric, so saving you valuable time and costs, when compared to classic on-premises proxy solutions that are typically DMZ bound, or rely on complex edge server topologies, all contributing to increased maintenance costs.

The applications in scope for being published via ’s Azure Active Directory Application Proxy deployment are listed below:

**Instruction:** please update the following table to illustrate which applications will be in this delivery.

**With Azure AD Application Proxy, you can access different types of internal applications:**

* Web applications that use Integrated Windows Authentication for authentication
* Web applications that use form-based access
* Web APIs that you want to expose to rich applications on different devices
* Applications hosted behind a Remote Desktop Gateway

|  |  |
| --- | --- |
| Application Name | Description |
| <Enter Application ame> | <Provide a brief description of the application |
| <Enter Application Name> | <Provide a brief description of the application |
| <Enter Application Name> | <Provide a brief description of the application |

Table 3: Applications to be published via Azure Active Directory Application Proxy

* 1. Core Dependencies

The following core dependencies are required to publish applications through Azure AD Application Proxy.

Note

The availability of services and features for a Microsoft cloud service varies by country or region. Features within a service can be restricted for legal reasons in certain countries or regions. To see if a service or feature is available with or without restrictions, look for your country or region on the service’s license restrictions site. Find more details at <https://msdn.microsoft.com/en-us/library/azure/dn499825.aspx>.

The following table illustrates the required components for this implementation.

|  |  |
| --- | --- |
| Component | Description |
| Active Directory Domain Services (on-premises directory) | Source of authority for <Update Customer Name in Doc Properties> identities and associated information. |
| Licensing | Azure subscription that supports Azure Active Directory App Proxy:   * A Basic subscription entitles you to App Proxy use only * A Premium I or EMS E3 subscription provides additional value adds such as MFA & Conditional Access, to name a few |
| Azure AD Connect | Used to synchronize <Update Customer Name in Doc Properties> identities and associated information from your on-premises AD DS environment(s) to the cloud, thus populating your Azure AD directory. |
| Connector hardware requirements | Capacity planning for the connector host should be based on any reasonably sized server capable of running Windows Server 2012 R2 or 2016.  Connector hosts can be either physical hardware or a VM run within any flavour of a hypervisor |
| Connector placement | A connector host should be able to connect to the Azure Application Proxy services in Azure, and the on-premises applications that you are publishing. In the interest of optimising the user experience, connectors should be positioned as close to their respective published backend applications, as possible.  Azure AD Application Proxy connectors cannot coexist on a host already running the Azure AD Pass-Through Authentication agent. |
| Connector network access requirements | Deployed connectors will attempt to connect to Azure over the following ports:   * 80: Used for downloading certificate revocation lists (CRLs) when validating the SSL certificates bound to our Azure endpoints * 443: All outbound communication with the Azure Application Proxy service are encrypted over TLS 1.2   A connector host’s Application Proxy service and its associated Updater service both run in the machine context, so would have no way of interacting with an outbound proxy that requires authentication. For this reason, we would recommend that all outbound traffic originating from a connector host is allowed through a forward proxy without being challenged.  Note that terminating of connector traffic is not supported and will inevitably the connector from establishing a secure channel with its associated tenant, and other key endpoints, so please avoid all forms of inline inspection on outbound communications to the WWW.  By the same token, load balancing of the Proxy connectors themselves is also not supported, or even necessary. |
| Connector domain join requirements | For single sign-on to your published applications using Kerberos Constrained Delegation, the connector host should be domain-joined in the same AD domain as the applications that you are publishing. For information, see KCD for single sign-on with Application Proxy. |

Table 4: Infrastructure Dependencies

* 1. Applications Discovery

This section details the applications discussed and the application insights during the kick off workshop, as part of the assessment phase.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Application Name | Application Platform: IIS, Apache, Tomcat, NGINX | Application Type: SharePoint, SAP, CRM, Custom, Rich Client, Citrix, RDG | Authentication Type: FBA, SAML, WIA, LDAP, Custom, OAuth | Current Remote Access Method |
| Application1 |  |  |  | F5 with MFA |
| Application2 |  |  |  | VPN |
| Application3 |  |  |  | UAG/TMG |

Table 5: External Prerequisites of Azure Active Directory Service.

* + 1. Application specific details

This section captures details for each of the applications to be published, as outlined in table 3.2, and should be used to profile all applications that are in scope for publishing via Azure AD Application Proxy.

Note

Azure AD Application Proxy supports publishing of web-based applications only

Web applications configured for client certificate authentication cannot currently be published through Azure AD Application Proxy. This includes SAP implementations that employ a dispatcher front end to pre-authenticate users

Please determine the following information, as appropriate:

|  |  |
| --- | --- |
| Application Properties | <Application Name> |
| Application Type | SharePoint, SAP, CRM, Custom Web Application, etc |
| Application Platform | Windows IIS, Apache, Tomcat, NGINX |
| Internal Access | * 1. What is the FQDN used to access the application internally?   2. Is the application accessed over HTTP or HTTPS internally (or both)?   3. Does the application draw content from sources other than itself?   4. If SharePoint, are AAMs configured? |
| Internal URL | 1. What is the internal FQDN used to access the application? |
| Internal network port | 1. Which TCP port(s) does the application use internally? |
| External URL | 1. What external FQDN do you want to use for external access? |
| Websockets details | 1. What websockets protocol and port is in use? |
| Public certificate | 1. Note subject name, serial #, and location of certificate that will be used to publish the application using a custom domain, in the Azure AD portal? |
| Authentication type | 1. Is the application accessed anonymously, or is it configured to authenticate users? 2. If so, what type of authentication is it configured for? e.g. Windows Integration Authentication, forms-based auth, header based authN, direct federation with an STS? 3. SPN/Kerberos auth questions. It feels too deep for here but if we’re going to do KCD we need to know the details and if not here then where? |
| Application Server Location | 1. Where is the web server/farm located on your network? Determining common network locations across the web applications is important as we will target Connector groups based on these. 2. Is the application front ended by a load balancer? |
| Web server domain? | 1. Domain Name? forest name? |
| Load balancing | 1. If a farm, what type of load balancing is in use? |
| Specific Requirements | 1. Any additional remote access requirements to take into account? |
| Users/Groups access | 1. Which users/groups of users should be able to access the application externally |
| Additional security controls | 1. Do you require any additional security controls for external access e.g. MFA or device-based CA? |
| Connector group name | 1. Define a logical name for the group of connectors that will provide the conduit and SSO to this application |

* 1. Pilot best practices recommendations

The process of making an application available externally through Azure AD Application Proxy is somewhat trivial, typically requiring very little effort and expertise. The amount of time and effort that is then required to fully commission an application with SSO, really depends on several external factors, including application type, its configuration, and available authentication options.

For this reason, we would perhaps suggest running a pilot on a standard IIS based web application that is already pre-configured for Integrated Windows Authentication(IWA), as this should require the least effort to successfully pilot remote access and SSO.

The following best practice elements should facilitate an optimal design for a pilot implementation.

* Procurement of a public certificate issued by a third party trusted certificate authority is a typically a better approach than using privately issued certificates. Azure Application Proxy supports standard, wildcard or SAN based certificates.
* If Kerberos Constrained Delegation is required to perform SSO, ensure that the Connector server is domain joined ahead of time, to avoid change control delays
* Existing connector instances should be of version 1.5.132.0 or higher, so please check this before publishing any applications, and upgrade if necessary
* Our recommendation is to install the Connector within the same domain as the service account that is used to run the backend web application
* Connectors need line of site to published web and domain controllers, if doing KCD SSO, so ensure to have all internal routing and firewall segregation changes raised ahead of time.
* Deploying a single connector for initial pilot testing is sufficient for proof testing and is often retained for ongoing staging of application publishing and more intricate uses cases. Web applications being piloted can later be moved into production connector groups
* Security is paramount, so SSL between the connector host and target applications should always be used. Particularly if the web application is configured for forms-based authentication (FBA), as user credentials are effectively transmitted in clear text.
* Your workforce is most likely to remember an external URL that has relevance, so try and avoid publishing your application using our pre-defined msappproxy.net or onmicrosoft.com suffixes. A much better approach would be to use a familiar top level verified domain, prefixed with a logical hostname E.g. intranet.<*customers\_domain*>.com
* Common internal and external URL. The URL used to address an application is critical to providing an optimal user experience and should be defined from the very outset. Where possible we would insist on using the same internal and external FQDN, as this offers multiple benefits, including:
* Links shared over email/IM will work regardless of user location
* Bookmarked URL’s will work regardless of location
* Native application performance can only be achieved in scenarios where the proxy is doing as little as possible. Link translation is an expensive process that requires parsing of all data in flight, as it’s being exchanged between a client and the backend applications, thus adding unnecessary load on the proxy service, whilst possibly affecting an applications ability to perform optimally. So, why not save the proxy tons of legwork by having the application itself return the same URL as used externally, by simply addressing it with that URL in the first place.
* When attempting to connect to a backend application, proxy connectors will normally perform a DNS query against the local site DNS servers. This is not the case if the proxy is being configured to route traffic via an internal forward proxy, as in that scenario we would expect the onwards proxy to handle all DNS resolution on behalf of clients.
* The use of localhost DNS records is perfectly fine for quick and dirty functional testing, before requesting internal DNS changes
* Where possible we would encourage the use of kerberos to achieve SSO, as this provides a more secure option to FBA, whilst also negating the use of our password vaulting browser plugin
* The closer a connector can be to published applications the better, as this will reduce round trip times and overall latency
* Limit visibility of the pilot web application’s icon in the Azure MyApps panel, to a pilot group of users. When ready for production you will be able to scope the app to its respective targeted audience, either in the same single tenant, or by also publishing the web application in your other, production tenant.
* Tools such as Fiddler for Windows or Charles for OSX will prove indispensable when troubleshooting issues with publishing applications. Unbeknown to many sysadmins, Fiddler can also act a proxy to help trace and debug traffic for mobile platforms such as iOS and Android, and pretty much anything that can be configured to route via a proxy.
  1. Production best practices recommendations

With a pilot successfully completed, you should now be able to move your published web applications into production.

The following best practice elements should facilitate an optimal design for a production implementation:

* To provide maximum high availability, connectors should be placed on hosts that have been designated for the sole purpose of handling Azure AD Application Proxy traffic only
* Grouping the web applications by geography/data location will allow for the most efficient means of end-to-end routing of traffic, and these can then be mapped to logically named Connector Groups. Please refer to the following [link](https://docs.microsoft.com/en-us/azure/active-directory/active-directory-application-proxy-connectors-azure-portal#use-cases-for-connector-groups) for placement and sample scenarios
* Azure Application Proxy provides sufficient levels of integrated security to fully protect published applications, but this often relies on security administrators having a fair understanding of how their applications work, to achieve an effective security posture. One simple method of maintaining a tight posture is to consider publishing applications at a path level, where you then only expose the exact folders and files that are required for the application to function, and no more. If doing this however, make sure that it includes all the necessary images, scripts, and style sheets for your application. For instance, if your app is at https://intranet.wacketywack.com/home/ but uses images located at https://intranet.wacketywack.com/media, then you should publish the root, https:// intranet.wacketywack.com/ as the path. Note that this internal URL doesn't have to be the landing page your users see. For more information on setting a custom home page for published apps, see [here](https://docs.microsoft.com/en-us/azure/active-directory/application-proxy-office365-app-launcher).
* If you are running a single tenant for test and production then migrating apps from pilot to production is simply achieved by relocating the production ready applications from the pilot connector, to the designated production connectors. Some organizations will run separate tenants however, one for pre-production and the other for production, and in that scenario, each will play host to a distinct namespace will that represents each environment, individually. If this is how you are setup then you will not have the ability to move piloted apps between tenants, so should ensure that the production tenant is also ready to host the apps. E.g. Validate that the required customer domain is already registered and verified, that external fqdns are registered in public DNS, and that SSL certificates are available to support any custom domains. The same SSL certificate types are supported, being standard, wildcard or SAN based, preferably issued by a 3rd party trusted certificate authority.
* As with the pilot, you should determine how the Connector server is expected to route out to the Azure App Proxy services. This can be direct access or via a forward proxy.
* Deploy Connector servers. A minimum of 2 Connector instances should be deployed per web application grouping to provide redundancy. The Connectors should be deployed in the same network segment as the published web servers and can run on physical hardware or a VM.
* If Kerberos Constrained Delegation is required for any of the web applications to be published through a Connector Group then the Connector servers must be domain joined, and our recommendation is to install the Connector in the same domain that the web application service account/machine accounts are in.
* Piloting applications will never provide a true reflection of the volume and traffic that an application is likely to be subjected to, once in production, so you may now want to carry out operational baselining to determine if further scale out is required beyond the initial 2 connector servers

1. Azure AD Application Proxy Deployment Plans

This section details the deployment steps required to publish the afore detailed applications via Azure AD Application Proxy, comprising of the following high-level activities:

|  |  |
| --- | --- |
| Section | Activity/Steps |
| 4.1 | Prerequisites |
| 4.2 | Use-case, scenario, capability requirements |
| 4.3 | Working with existing on-premises proxy servers |
| 4.4 | Enable Application Proxy and deploy connectors |
| 4.5 | Create connector groups |
| 4.6 | Publish applications |
| 4.7 | Functional Testing |
| 4.8 | Enable single-sign on and SSO |
| 4.9 | Working with “Other” applications |
| 5.0 | Enable conditional access |
|  |  |

Table 7: List of tasks to implement Azure AD Application Proxy

* 1. Prerequisites

Infrastructure changes can take time to go through change control, so getting all pre-requisites arranged way ahead of time is key to a smooth and timely deployment.

Before continuing you should have completed the following activities:

* Licensing in place
* Application discovery
* Custom domains [verified in tenant](https://docs.microsoft.com/en-us/azure/active-directory/active-directory-domains-add-azure-portal)
* CNAME record created in public DNS for each application to be published
* Certificate(s) procured
* Assessed Connector Group/Placement
* Connector Network access requirements – Internal
* Connector Network access requirements – External
* Connector domain join requirements for single-sign on
  1. Use-case/scenario/capability requirements

The below are the most common use cases for Azure AD Application Proxy deployments.

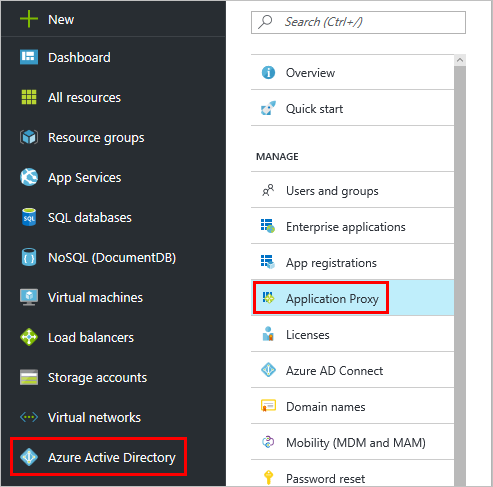
* Domain users will need to access the application securely with seamless single-sign on from on-premises from any domain joined or Azure AD joined devices.
* Azure AD users will need to access the application securely with seamless single-sign on from on-premises from any domain joined or Azure AD joined devices.
* Users with BOYD approval can securely access the application provided they have enrolled from MFA and that they have registered Microsoft Authenticator app as an authentication method and a mobile phone.
* Additional use-case/scenario/capability requirements go here…
  1. Working with existing on-premises proxy servers

Having connectors route via an outbound proxy when connecting to our Azure App Proxy cloud, is possible, and complete guidance on this topic can be found [here](https://docs.microsoft.com/en-us/azure/active-directory/application-proxy-working-with-proxy-servers).

What’s important to note however, is that limiting these outbound connections to specific Azure target IPs is not advisable, as the IP addresses that host our Azure services can span many hundreds of ranges in order to provide a highly available and elastic cloud service, that can often change to meet certain demands, at any given time. For reference, all address and IP information pertaining to our Azure cloud services can be found in our [Azure Datacenter IP ranges](https://www.microsoft.com/en-us/download/details.aspx?id=41653) document, which incidentally gets updated weekly, so would make it very challenging for proxy and firewall admins to keep updating your outbound ACLs to coincide with our changes. Not to mention that any overlooked changes could potentially affect a connectors ability to connect to our cloud, thereby causing unforeseen impact, so instead our advice would be to consider either of the following:

* Limit outbound connector traffic to specific fqdns, but this capability is wholly dependent on your outbound proxy device offering such functionality
* Or simply consider allowing HTTP & HTTPS from connector hosts out to ‘Any location, and lock down each host so that general user browsing is not possible
  1. Enable Application Proxy and Deploy Connectors

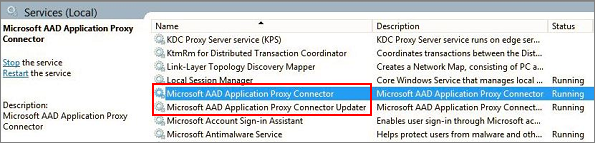
1. Sign into the [Azure portal](https://portal.azure.com) using a Global Administrator account
2. Your current directory appears under your username in the top right corner. If you need to change directories, select that icon.
3. Head to the **Azure Active Directory** > **Application Proxy** blade



1. Select **Download Connector**

Download Connector

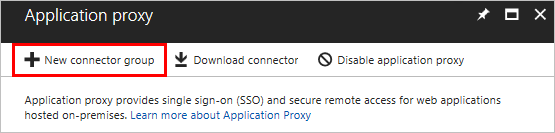
1. Run **AADApplicationProxyConnectorInstaller.exe** on the server you prepared per the prerequisites.
2. Follow the instructions in the wizard to run through the installation
3. During installation, you are prompted to register the connector with the Application Proxy of your Azure AD tenant.
   * Provide your Azure AD global administrator credentials. Your global administrator tenant may be different from your Microsoft Azure credentials.
   * Make sure the admin who registers the connector is in the same directory where you enabled the Application Proxy service. For example, if the tenant’s domain is contoso.com, the admin should be globaladmin@contoso.com, or any other alias on that domain.
   * If **IE Enhanced Security Configuration** is set enabled on the server where you are installing the connector, the registration screen might be blocked. Follow the instructions in the error message to allow access, to disable Internet Explorer Enhanced Security.
   * If connector registration does not succeed, click on the Troubleshoot Application Proxy link in the wizard, or use the following [link](https://aadap-portcheck.connectorporttest.msappproxy.net/)
4. When the installation completes, two new services are added to your server:
   * **Microsoft AAD Application Proxy Connector** enables connectivity
   * **Microsoft AAD Application Proxy Connector Updater** is an automated update service, which periodically checks for new versions of the connector and updates the connector as needed.



1. Deploying additional connectors is as easy as repeating steps 2 and 3, where each connector will be registered individually.
   1. Create Connector Groups

You can create as many connector groups as you want. Connector group creation is accomplished in the [Azure portal](https://portal.azure.com).

1. Select **Azure Active Directory** to go to the management dashboard for your directory. From there, select **Enterprise applications** > **Application proxy**.
2. Select the **Connector Groups** button where the New Connector Group blade appears
3. Click on the **New Connector Group** button



1. Give your new connector group a name, then use the dropdown menu to select which connectors belong in this group.
2. Select **Save** when your connector Group is complete.
   1. Publish Applications

There are many approaches and options for publishing applications, not all of which can be covered in this paper, so this section aims to cover some of the most fundamental areas of setting up applications for external access.

Before doing so, the assumption is that you would have already satisfied all of the pre-requisites, and deployed several connectors that are now showing as active in the **Azure Active Directory** > **Application Proxy** blade.

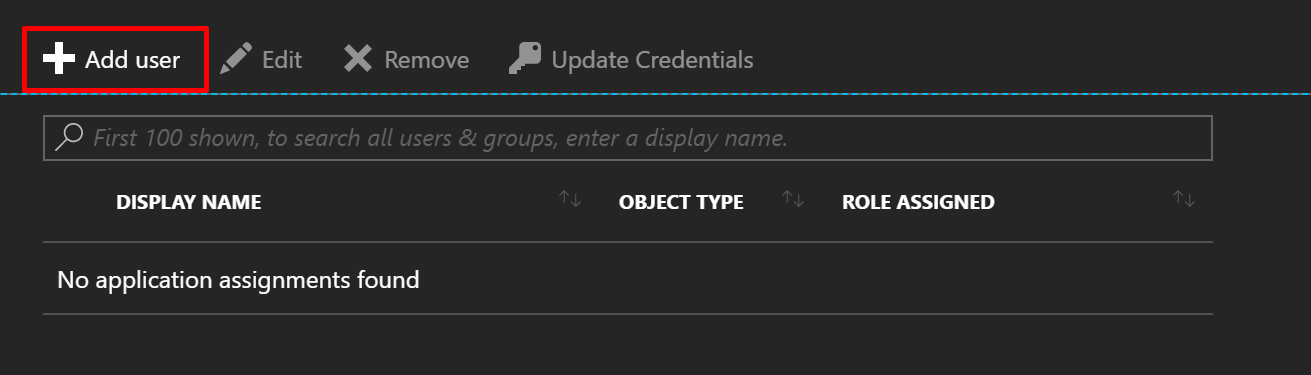
## Choosing the right Applications type to publishAssign applications to your connector groups

The last step before testing is to assign your application to a connector group.

1. Under **Connector group**, use the dropdown menu to select the group you want the application to use.
2. Select **Save** to commit the change

## Assigning Users to your Application

1. Still in the application’s blade, select **Users and Groups and click on Add user**



Congratulations, your application is now published and should be accessible by typing its external URL directly in a browser, or logging into <https://myapps.microsoft.com> and finding its icon there.

* 1. Base Functional Testing of Applications

Above all, our advice is always to try and keep things as simple as possible, and build up from there. For instance, you could start by validating that a published application’s content *is* entirely accessible from the WWW, without the added complexity of enabling Azure AD pre-authentication or SSO. This would help validate that all ground work is in place, and that all links working, but equally that your configurations are sound, too. I.e…

1. Test and validate access to the web application, with pre-authentication and SSO disabled.
2. Enable pre-authentication if required and assign Users/Groups. Test and validate access.
3. Add authentication delegation if required. Test and validate access.
4. Apply Conditional Access / MFA policies as required. Test and validate access.

Running multiple tenants will involve publishing applications in pilot tenancy and then in the production tenant, and go through the exact same functional test sequence performed in pilot, before commissioning the applications

* 1. Pre-Authentication and Single Sign On

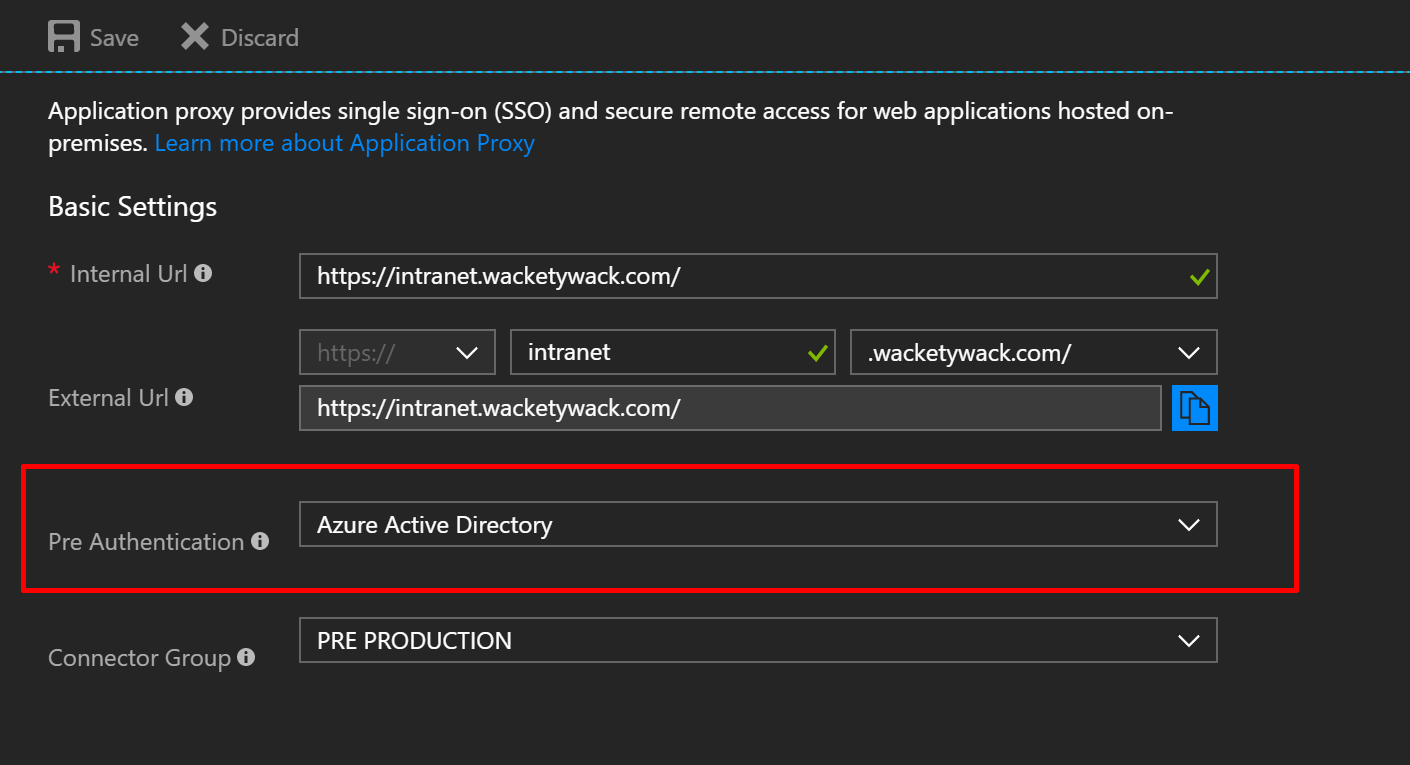
Single sign-on is a key element of Azure AD Application Proxy value proposition, as it provides the best possible user experience, through users only having to sign in the once, to Azure Active Directory. Once they authenticate to Azure Active Directory, the Application Proxy connector handles the authentication to the on-premises application, on behalf of the user. In this scenario the backend application is completely oblivious to the SSO operation and processes the login as if it were the user themselves.

That said it should be noted that the ability to do SSO is only possible if Azure AD has knowledge of the user requesting access to a resource however, so a pre-requisite for SSO is that your application is configured to pre-authenticate users, upon access. Or in other words, selecting Passthrough would allow users to access the published application without ever having to authenticate to Azure Active Directory, so could Azure AD possibly perform “on behalf of” SSO if it has no user identity send down to the connector.

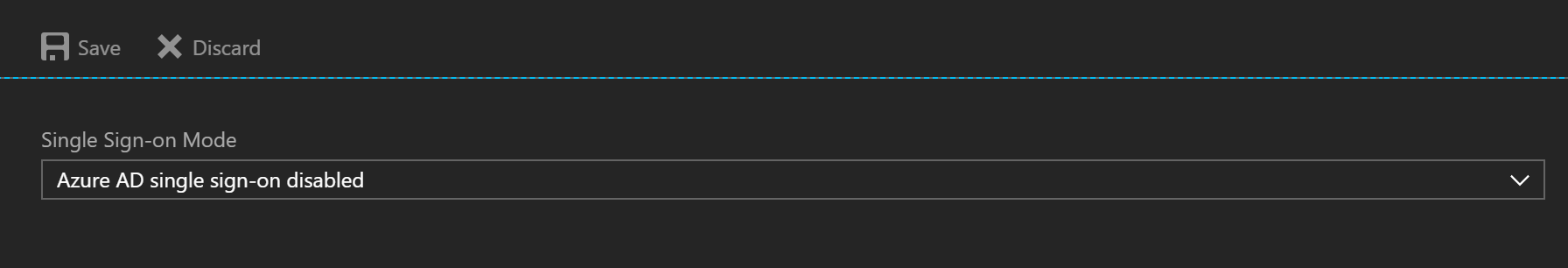
So, on that basis let’s go ahead and take our first step towards enabling SSO.

## Enabling Pre-Authentication

1. Navigate to **Azure Active Directory** > **Enterprise applications** > **All applications** and choose the app you want to manage.
2. Select **Application Proxy**.
3. In the Pre Authentication field, use the dropdown list to select **Azure Active Directory**, and hit **Save**



It’s probably worth mentioning that this will also leave SSO disabled, in the Single Sign-On blade…



So, you should take the opportunity to verify your application is still accessible through the Application Proxy, before moving onto the next section. The expected behavior is that Azure AD will challenge you for authentication and then the backend application should also do the same.

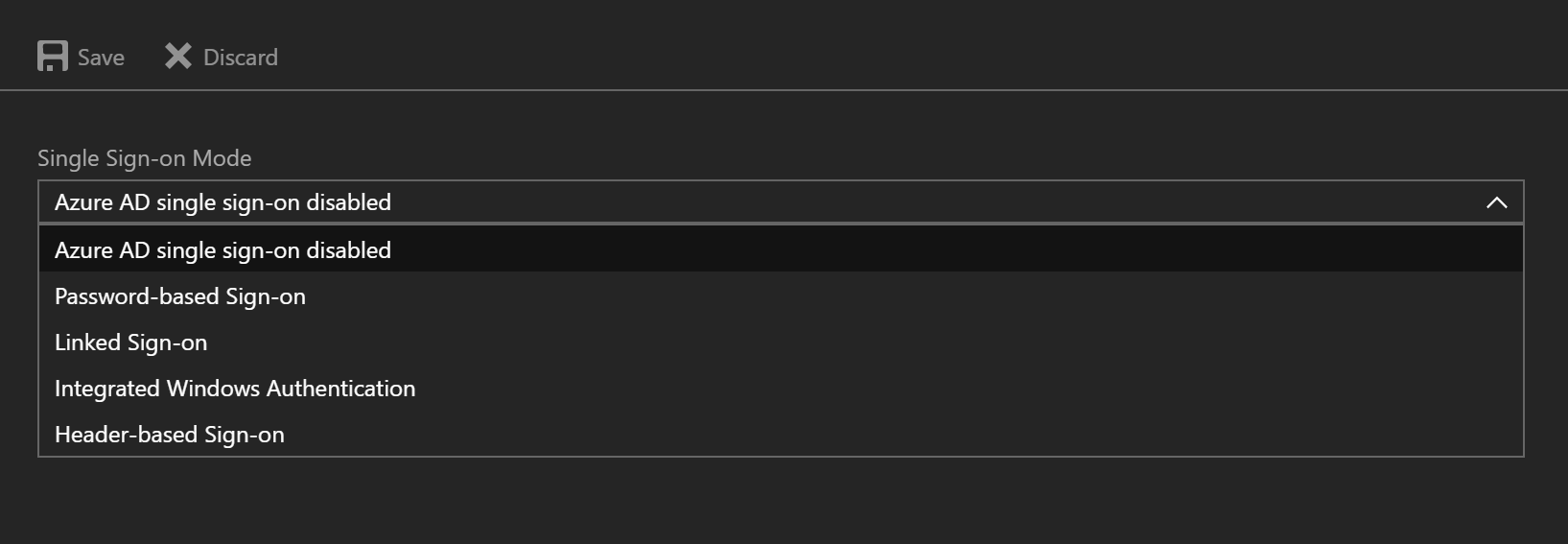
## Enabling Single Sign-On

With your application secured it’s time to look at the user experience, and what better to way to put a spring in every user’s step, than Single Sign-On itself!

This section gets into the more intricate side of publishing applications, and can vary in complexity, depending on application type, and the options it offers for authentication. Fortunately however, our Azure AD Application Proxy feature team have done a sterling job in making Azure AD Application Proxy as agnostic as one would expect for a cloud service, so our options for performing SSO should be sufficient enough to accommodate most applications.

1. Navigate to **Azure Active Directory** > **Enterprise applications** > **All applications**.
2. Select the app whose single sign-on options you want to manage.
3. Select **Single sign-on**.

The dropdown menu shows five options for single sign-on to your application:



## So, what do all of these options do, exactly…

## **Azure AD single sign-on disabled**

With this option selected, your users may authenticate twice. First, to Azure Active Directory and then to the backend application itself. This option is a good choice for exposing anonymous authentication applications out on the WWW, but does offer the option of enabling pre-authentication, in scenarios where you may want to constrain access to corporate users only.

## [**Password-based sign-on**](https://docs.microsoft.com/en-us/azure/active-directory/application-proxy-sso-azure-portal)

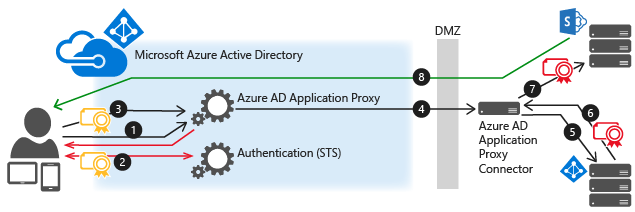
This option is a good choice for providing SSO to applications that are limited to authenticating using forms-based authentication. In this scenario your users will be expected to manually sign in to the application the first time they access it, and after that, Azure Active Directory will take care of supplying the username and password on behalf of the user, through a browser add-on.

## [**Linked sign-on**](https://docs.microsoft.com/en-us/azure/active-directory/active-directory-appssoaccess-whatis#how-does-single-sign-on-with-azure-active-directory-work)

Choosing this option tells Azure AD Application Proxy to perform pre-authentication, but to then hand off application authentication to another existing single sign-on solution. In this scenario the user is redirected to another STS that will handle token issuances, before eventually redirecting the user back to the backend application.

## [**Integrated Windows Authentication**](https://docs.microsoft.com/en-us/azure/active-directory/active-directory-application-proxy-sso-using-kcd)

This option makes the most sense when your on-premises applications are already configured for Integrated Windows Authentication(IWA), and capable of kerberos authentication. With this option, your users only need to authenticate to Azure Active Directory, and then our Application Proxy connector impersonates the user to get a Kerberos token for the application, using Kerberos Constrained Delegation(KCD).



## [**Header-based sign-on**](https://docs.microsoft.com/en-us/azure/active-directory/application-proxy-ping-access)

This is the least common approach at doing SSO, in scenarios where your application performs authentication/authorization, through the use of headers. In this scenario your users only need to pre-authenticate to Azure Active Directory, and a middle ware component provided by a 3rd party partner, Ping, then handles injection of access tokens into a header format for the application to consume.

The option most appropriate for you applications will ultimately depend on what the application offers, so please refer to the application details section for full information on what your applications support, as this will ultimately dictate how we do SSO.

Et viola! You can now take start layering up with any other controls and capabilities that may be required to further secure access to your application.

* 1. Working with “Other” Applications

In addition to web browser based applications, Azure Active Directory Application Proxy can also be used to publish other types of client applications that have been developed to leverage our Azure AD Authentication Library ([ADAL](https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-authentication-libraries)), or Microsoft Authentication Library([MSAL](https://azure.microsoft.com/en-us/blog/start-writing-applications-today-with-the-new-microsoft-authentication-sdks/)).

In these scenarios, Application Proxy supports native client apps by consuming Azure AD issued tokens that are received in the header information of client request, allowing our Application Proxy service to perform pre-authentication on behalf of the users. This solution does not use application tokens for authentication.

Details on publishing native client apps are available [here](https://docs.microsoft.com/en-us/azure/active-directory/active-directory-application-proxy-native-client), and another useful link for claims based applications can be found [here](https://docs.microsoft.com/en-us/azure/active-directory/active-directory-application-proxy-claims-aware-apps).

* 1. Enforcing Conditional Access

Traditional security solutions used to be enough to protect your business. But that was before the mobility industry grew, which created a larger attack landscape, and the transition to the cloud made employees' interactions with other users, devices, apps, and data more complex. To truly protect your business now, you need to take a more holistic and innovative approach to security, one that can protect, detect, and respond to threats of all kinds on-premises as well as in the cloud.

In more than 63 percent of data breaches, attackers gain corporate network access through weak, default, or stolen user credentials. Microsoft Identity-Driven Security focuses on user credentials, slamming the door shut on credential theft by managing and protecting your identities, including your privileged and non-privileged identities.

User- and location-based conditional access

Keep sensitive data protected by limiting user access based on geo-location or IP address with [location-based conditional access policies](https://docs.microsoft.com/azure/active-directory/active-directory-conditional-access-locations).

Device-based conditional access

Ensure only enrolled and approved devices can access corporate data with [device-based conditional access](https://docs.microsoft.com/azure/active-directory/active-directory-conditional-access-policy-connected-applications).

Application-based conditional access

Work doesn't have to stop when a user is not on the corporate network. [Secure access to corporate cloud and on‑premises apps](https://docs.microsoft.com/azure/active-directory/active-directory-conditional-access-mam) and maintain control with conditional access.

Risk-based conditional access

## Protect your data from malicious hackers with a [risk-based conditional access policy](https://www.microsoft.com/en-us/cloud-platform/conditional-access) that can be applied to all apps and all users, whether on-premises or in the cloud.