IdentityIQ Microsoft Teams Notifications

Version: 8.4

Revised: September 2023

Copyright and Trademark Notices

Contents

Introduction to IdentityIQ Microsoft Teams Notifications

IdentityIQ's Microsoft Teams Notifications feature allows notifications from IdentityIQ to be delivered directly to users as notifications in their Microsoft Teams environment.

Any IdentityIQ notification for certifications, access requests, and access approvals can be sent as a Microsoft Teams notification, and can include a link to the relevant area or action within IdentityIQ. For more information on IdentityIQ notifications, see the IdentityIQ **System Configuration** documentation.

With SAML-based single sign-on enabled, authenticated Microsoft Teams users can directly access IdentityIQ from links contained within the notifications, to take actions in IdentityIQ, without a separate login to IdentityIQ.

Configuring IdentityIQ Microsoft Teams Notifications

Implementing the Microsoft Teams Notifications feature for IdentityIQ involves creating and configuring some components in Azure, and also configuring IdentityIQ to support Microsoft Teams notifications. Information from some configuration steps is required by other configuration steps, so it is strongly recommended that you perform the integration in the sequence presented in this guide.

Configuring SAML-based single sign-on (SSO) between Azure and IdentityIQ is an *optional* step in this process. Enabling SSO between IdentityIQ and Azure gives your Microsoft Teams users a seamless experience using notifications in Microsoft Teams to take action in IdentityIQ. Enabling SSO removes the need to log in separately to IdentityIQ or to browse to the correct IdentityIQ location to take action on a notification.

> Important: Details on how to configure Azure components are provided in this guide as an aid to implementers; however, implementers should also consult Microsoft's documentation on Azure and Microsoft Teams to ensure that they have the most accurate and up-to-date information on these platforms. This guide only discusses actions in Azure that are required as part IdentityIQ's Microsoft Teams Notifications feature, and does not discuss more general Azure concepts or actions that may be part of setting up Microsoft Teams and SSO for your organization.

Prerequisites for Integrating Microsoft Teams with IdentityIQ

Prerequisites for implementing the Microsoft Teams integration for IdentityIQ include the following:

Supported Operating Systems

IdentityIQ's Microsoft Teams bot is supported on any currently available and supported versions of Windows and Linux.

Azure Prerequisites

- A familiarity with Azure concepts and operations on the part of your implementation team
- A configured and functioning Azure tenant for your organization
- A configured IdentityIQ Connector application. This application provides the permissions necessary for aggregating Active Directory user and group data into IdentityIQ. It communicates with a corresponding Azure Active Directory application within IdentityIQ. For more information, see SailPoint's *Integrating SailPoint with Azure Active Directory Connector* guide, which is available in the **Microsoft Azure Active Directory** area of the Connector Directory on Compass.

- *Optional*: a resource group in Azure for your IdentityIQ integration. This can help with organizing your Azure resources.

IdentityIQ Prerequisites

- A configured instance of an Azure Active Directory application. This is used to aggregate user and group data from Azure Active Directory into IdentityIQ, and correlate it to IdentityIQ users. Some modifications will be made to this application as part of setting up the Microsoft Teams integration, and are described later in this document. You can also refer to SailPoint's *Integrating SailPoint with Azure Active Directory Connector* guide, which is available in the **Microsoft Azure Active Directory** area of the Connector Directory on Compass.
- *Recommended*: a private server to run the IdentityIQ service code. The IdentityIQ service code should **not** be run on the same server as IdentityIQ; the service code must be exposed to the Internet, and the IdentityIQ server should **not** be exposed to the Internet.

Connectivity and Security Prerequisites

- A network connection that allows Microsoft Teams to contact the IdentityIQ service code. This requires a public IP address that routes to the private IP of the server running the service code. The traffic can be limited to a specific port, which is customizable. The public IP must be resolvable by a DNS.
- To enable Microsoft Teams to trust the call to the IdentityIQ service code, it must support HTTPS and have a valid certificate from a Microsoft supported Certificate Authority for the domain name in the previous item. The certificate and key are used by the IdentityIQ service code running on the private server. Note that Microsoft Teams will not send messages to a server with an invalid certificate; this includes a self-signed certificate.

Components of IdentityIQ's Microsoft Teams Notifications

Here is an overview of the components that are part of the IdentityIQ Microsoft Teams Notifications feature. Some of these components are created as **app registrations** in Azure. Detailed instructions for creating and configuring these components is provided later in this document.

API Application – an Azure app registration that enables token-based authentication between Microsoft Teams and IdentityIQ by providing a `GetToken` scope and the proper audience value. This will be assigned to the Teams application as a permission. When this is enabled, IdentityIQ can look for the scope `GetToken` when the bot requests an API token, and verify the audience token claim. See Creating an API Application in Azure.

Teams Application – an Azure app registration that defines the scope in Azure that can be used by the IdentityIQ Microsoft Teams Notifications feature. In this app, you define authentication parameters, set a client secret, and choose which Microsoft Graph API permissions to expose. You will use some values from this app in IdentityIQ, to ensure secure communication between IdentityIQ and Azure. See Creating a Microsoft Teams Application for IdentityIQ in Azure

Azure Bot for Microsoft Teams – an Azure bot that leverages Microsoft's artificial intelligence (AI) chatbot to manage the messaging between Microsoft Teams and IdentityIQ. This bot also stores the configuration required to access the IdentityIQ service code. See Creating an Azure Bot for IdentityIQ's Microsoft Teams Notifications.

IdentityIQ service code – this code is provided by SailPoint and is installed in your environment. The service code package includes an environment (`.env`) file that must be configured with necessary resource information such as the Azure Teams app ID, Azure Teams app secret, Azure tenant ID, and more. Once the `.env` file has been configured, a manifest can be built. This manifest is used to deploy the app in the user's Teams environment. See Installing and Configuring the IdentityIQ Service Code

Optional: **SSO Provider** – an Azure app registration that enables SAML-based single-sign on between Microsoft Teams and IdentityIQ. Implementing this option streamlines the login process for Microsoft Teams users, allowing them bypass the IdentityIQ login when they click IdentityIQ links in Microsoft Teams. See Configuring Single Sign-On to IdentityIQ from Microsoft Teams

Best Practices for Configuring IdentityIQ Microsoft Teams Notifications

SailPoint recommends that you follow these best practices as you implement the IdentityIQ Microsoft Teams Notifications feature:

- Use **meaningful names** for the components you create in Azure. This will help with organizing your applications in Azure. Here is an example of how you could name your components:
    - **IdentityIQ API Access** – the Azure API application. See Creating an API Application in Azure.
    - **IdentityIQ Teams** – the Teams application. See Creating a Microsoft Teams Application for IdentityIQ in Azure
    - **IdentityIQ_MyCompany Bot** – the bot handle for your Azure bot. Note that this name must be **unique** in Azure. See Creating an Azure Bot for IdentityIQ's Microsoft Teams Notifications
    - **IdentityIQ Connector** – the Azure application for the IdentityIQ Azure Active Directory connector. See Prerequisites for Integrating Microsoft Teams with IdentityIQ.
    - **IdentityIQ SSO Provider** – the Azure application for managing single sign-on between Microsoft Teams and IdentityIQ. See Configuring Single Sign-On to IdentityIQ from Microsoft Teams.
- When **configuring the IdentityIQ service code**:

- Encrypt the certificate and key. See Installing and Configuring the IdentityIQ Service Code.
- Make sure the `cert` directory, the `bot.cert` and the `bot.key` items are set with `read write` permission for owner only.

Configuring Single Sign-On to IdentityIQ from Microsoft Teams

IdentityIQ Microsoft Teams notifications supports a SAML-based Single Sign-On (SSO) configuration. Enabling SSO for Microsoft Teams and IdentityIQ is *optional*, but it is useful in that it streamlines the login process for Microsoft Teams users, allowing them bypass the IdentityIQ login when they click IdentityIQ links in Microsoft Teams.

This is a two-step process that requires configuring SAML SSO within Azure for IdentityIQ, then using some of the parameters from the Azure steps to enable SAML SSO within IdentityIQ.

Single Sign-On Configuration in Azure Active Directory

> Details on how to configure Azure components are provided in this guide as an aid to implementers; however, implementers should also consult Microsoft's documentation on Azure and Microsoft Teams to ensure that they have the most accurate and up-to-date information on these platforms. This guide only discusses actions in Azure that are required as part IdentityIQ's Microsoft Teams Notifications feature, and does not discuss more general Azure concepts or actions that may be part of setting up SSO or Microsoft Teams for your organization.

Follow these steps in Azure for enabling single sign-on from Microsoft Teams to IdentityIQ:

1. Navigate to your Azure Active Directory home.
2. Click **Add > Enterprise Application**.
3. Click **Create your own application**
4. Enter a **Name** for your application.
5. Select **Integrate any other application you don't find in the gallery**.
6. Click **Create**. Once the creation process is complete, Azure opens a **Properties** page for the application.
7. Define the users and groups that can use SAML SSO with IdentityIQ:
    a. In the left navigation, click **Users and groups**.
    b. Click **Add users and groups**.

c. Choose the users and groups that you want to be able to use SAML SSO.

d. Click **Select**.

e. Click **Assign**.

8. In the left navigation, click **Single sign-on**.

9. Click the **SAML** tile

10. In the **Basic SAML Configuration** section, click **Edit**.

    a. Add an **Identifier (Entity ID)**. This can be any name, but typically takes the form of a URL (for example, `https://myhost:myport/identityiq`). This value is used to link Azure and IdentityIQ; Make a note of it to use later when configuring SSO within IdentityIQ.

    b. Add a **Reply URL (Assertion Consumer Service URL)** and enter a URL to your IdentityIQ instance, in this format:

    `https://myhost:myport/identityiq/login.jsf`.

    This URL must be accessible to your Microsoft Teams users on your network.

    c. Enter a **Sign on URL** to your IdentityIQ instance, in this format:

    `https://myhost:myport/identityiq`

    d. Do not enter a **Logout URL**.

    e. **Save** your changes.

11. In the **User Attributes and Claims** section, click **Edit**. IdentityIQ uses the unique identifier that is specified as the **Required Claim's Claim Name** to correlate IdentityIQ users to Azure/Microsoft Teams users. By default, Azure uses `user.userprincipalname` as its unique identifier, but IdentityIQ uses `user.objectid` by default.

    To use IdentityIQ's default configuration, you must edit this value to change the **Source attribute** for the claim to `user.objectid`.

    To edit the claim to use `user.objectid` as the **Source attribute**, click **Edit**.

a. Click the **Claim Name** to edit it.

b. Change the **Name Identifier format** to **Unspecified**.

c. Select `user.objectid` as the **Source attribute**.

d. **Save** your changes.

12. In the **SAML Signing Certificate** section, download and save the **Certificate (Base64)**. You will need this later, when configuring SAML SSO within IdentityIQ.

13. In the **Set up *(name)*** section, make a note of the URLs that are listed. You will need these later, when configuring SAML SSO within IdentityIQ.

Single Sign-On Configuration in IdentityIQ

Follow these steps for enabling single sign-on in IdentityIQ:

1. Click the **gear menu > Global Settings > Login Configuration**.

2. Click the **SSO Configuration** tab.

3. Check to select the **Enable SAML Based Single Sign-On (SSO)** box. This opens a panel for entering SAML Based SSO details.

4. In the **Identity Provider Settings** section:

   a. Enter an **Entity ID / Issuer**. This must match the **Azure AD Identifier** from the **Set up *(name)*** section in Azure. (See step 13 in the previous section.)

   b. Enter a **SSO Login URL**. This must match the **Login URL** from the **Set up *(name)*** section in Azure. (See step 13 in the previous section.)

   c. Enter your **Public X.509 Certificate**. This is the certificate you downloaded and saved in the Azure **SAML Signing Certificate** panel, as described the previous section.

5. In the **Service Provider (IdentityIQ) Settings** section:

   a. Enter an **Entity ID Issuer**. This must match the **Identifier (Entity ID)** value you set in Azure in the **Basic SAML Configuration** panel, as described in the previous section.

   b. Enter a **SAML URL**. This must match the **Reply URL (Assertion Consumer Service URL)** you set Azure in the **Basic SAML Configuration** panel, as described in the previous section.

c. Set the **SAML Name ID Format** to **unspecified**, if you are using the default identifiers as described in step 11 in the previous section. If you have opted to use a different set of values for uniquely identifying users, select the appropriate value here.

d. The **SAML Correlation Rule** manages the correlation of IdentityIQ identities to Azure/Teams identities. IdentityIQ provides a sample rule, **ExampleAzureActiveDirectorySAML**. This rule is included in the `examplerules.xml` file, located in the `[install_directory]WEB-INF\config` directory. If you are using the recommended default identifiers, you can use the sample rule without modification. If you have opted to use different unique identifiers, you can either edit this rule, or use a different rule that you have developed.

6. **Save** your changes.

For more information on configuring your IdentityIQ instance for SSO authentication, refer to the **SSO Configuration** section of the *SailPoint IdentityIQ System Configuration Guide*

Creating an API Application in Azure

To enable token-based authentication between Microsoft Teams and IdentityIQ, you must create an API application in Azure, then use some of the values from the API application within IdentityIQ, to enable authentication. For related information about what to configure in IdentityIQ for token-based authentication, see Configuring API Authentication for Microsoft Teams in IdentityIQCreating an API Application in Azure

To create the API Application in Azure:

1. Navigate to your **Azure Active Directory** home.
2. In the left navigation, click **App registrations**.
3. Click **New Registration**.
4. Enter a **Name** for your application. When creating applications, it's a good idea to name them according to function; for example, `IdentityIQ API`.
5. Under **Supported account types**, choose the **Multitenant** option.
6. Click **Register**
7. Once the registration process has completed, click **Expose an API**, in left navigation.
8. Click **Add a scope**
9. Click **Save and Continue**

10. Enter a **Scope name**, for example, `GetToken`. You will use this value later when configuring API Authentication in IdentityIQ.
11. For **Who can consent?** choose **Admins and users**
12. Add names and descriptions for **Admin consent** and **User consent**.
13. Make sure the state is **Enabled**
14. Click **Add scope** to save your information
15. Click to edit the **Application ID URI**, and append `/identityiq/api` to the end of the URI, then click **Save**. You will use this value later when configuring API Authentication in IdentityIQ.

Creating a Microsoft Teams Application for IdentityIQ in Azure

This Azure application defines the scope that can be used by the IdentityIQ Microsoft Teams Notifications feature. In this app, you define authentication parameters, set a client secret, and choose which Microsoft Graph API permissions to expose. You will use some values from this app in IdentityIQ, to ensure secure communication between IdentityIQ and Azure.

> Important: Details on how to configure Azure components are provided in this guide as an aid to implementers; however, implementers should also consult Microsoft's documentation on Azure and Microsoft Teams to ensure that they have the most accurate and up-to-date information on these platforms. This guide only discusses actions in Azure that are required as part IdentityIQ's Microsoft Teams Notifications feature, and does not discuss more general Azure concepts or actions that may be part of setting up Microsoft Teams and SSO for your organization.

1. Navigate to your **Azure Active Directory** home.
2. In the left navigation, click **App registrations**.
3. Click **New Registration**.
4. Enter a **Name** for your application. When creating applications, it's a good idea to name them according to function; for example, `IdentityIQ Teams`.
5. Under **Supported account types**, choose the **Multitenant** option.
6. Click **Register**.
7. Configure an Authentication redirect URI:

a. In the left navigation, click **Authentication**.

b. Click Add platform.

c. In the **Configure Platforms** panel that opens at right, click the **Web** tile.

d. Enter a **Redirect URIs**. Use this value for the redirct:
`https://token.botframework.com/.auth/web/redirect`

e. Under **Implicit grant and hybrid flows**, make sure both flows are checked.

f. **Save** your edits.

8. Create a client secret for the application:

a. In the left navigation, click **Certificates and secrets**.

b. Click **New client secret**.

c. Add a **description** and **expiration date**.

d. Click **Add**.

> **IMPORTANT**: This is the only time the client secret is available to be copied. Use the copy icon to **copy and save the secret.** You will need the secret later, when creating your Azure bot, configuring the service code, and configuring IdentityIQ's integration properties.

9. Configure the API permissions for the application:

a. In the left navigation click **API permissions**.

b. Click **Add a permission**.

c. Under the **Microsoft APIs** tab, choose **Microsoft Graph** and select these options in the **Delegated Permissions** list:

- `email`
- `offline_access`

- `openid`
- `profile`

Then click **Add permissions**.

> Note: If `User.Read` is present and selected, you can leave it selected.

d. In the left navigation click **API permissions**.

e. Click **Add a permission**.

f. Under the **My APIs** tab, choose the **API Application** you created earlier (as described in the Creating an API Application in Azure section). Select your `GetToken` permission and click **Add permissions**.

g. The API permissions page now shows a list of the API permissions you have created. Check the **Grant admin consent for (*MyTenant*)**, where *MyTenant* is the name of your tenant. Then click **Yes** to confirm.

10. In the left navigation, click **Expose an API** .

11. Click **Set** for the **Application ID URI**.

12. Edit the URI to add `botid-` immediately after the `api://` in the URI. Be sure to include the dash after `botid`. Click **Save**.

13. Add an **access_as_user** scope:

    a. Click **Add a scope**. Give it a Scope name and choose Admins and users. Provide Admin and user content descriptions – these are messages that will appear to users if they need to give authorization.

    b. Enter a **Scope name**, for example, `access_as_user`.

    c. Add names and descriptions for **Admin consent** and **User consent**.

    d. Make sure the state is **Enabled**.

    e. Click **Add Scope** to save your changes.

14. Add a **GetToken** scope:

a. Click **Add a scope**. Give it a Scope name and choose Admins and users. Provide Admin and user content descriptions – these are messages that will appear to users if they need to give authorization.

b. Enter a **Scope name**, for example, `GetToken`.

c. Add names and descriptions for **Admin consent** and **User consent**.

d. Make sure the state is **Enabled**.

e. Click **Add Scope** to save your changes.

15. Authorize desktop and mobile applications to use this application.

a. Click **Add a client application**

b. For a Microsoft Teams web client, enter this value as the **Client ID**:
`5e3ce6c0-2b1f-4285-8d4b-75ee78787346.`
This value is provided by Microsoft; refer to Microsoft's documentation for details about the Client ID.

c. Check the **Authorized scopes** option.

d. Click **Add application**.

e. Repeat steps 1-4 above, substituting this value in step 2: For a Microsoft Teams mobile and desktop clients, enter this value as the **Client ID**:
`1fec8e78-bce4-4aaf-ab1b-5451cc387264.`
This value is provided by Microsoft; refer to Microsoft's documentation for details about the Client ID.

Once your Microsoft Teams application is configured, **copy and save the Application (client) ID value**. You will need this value later, when creating your Microsoft Teams bot and configuring IdentityIQ's integration with Microsoft Teams.

For more information, see:

- Creating an Azure Bot for IdentityIQ's Microsoft Teams Notifications
- Configuring IdentityIQ's Microsoft Azure Active Directory Application for Teams

Creating an Azure Bot for IdentityIQ's Microsoft Teams Notifications

Next you will create an Azure bot that leverages Microsoft's artificial intelligence (AI) chatbot to manage the messaging between Microsoft Teams and IdentityIQ. This bot also stores the configuration required to access the

IdentityIQ service code.

> Important: Details on how to configure Azure components are provided in this guide as an aid to implementers; however, implementers should also consult Microsoft's documentation on Azure and Microsoft Teams to ensure that they have the most accurate and up-to-date information on these platforms. This guide only discusses actions in Azure that are required as part IdentityIQ's Microsoft Teams Notifications feature, and does not discuss more general Azure concepts or actions that may be part of setting up Microsoft Teams and SSO for your organization.

1. From your Azure home, use the search field to search for **Azure Bot**.
2. Choose the result that appears in the **Marketplace** category, to open the **Create an Azure Bot** page.
3. Enter a **Bot handle**. This is the name of your bot, that your users will see in the Microsoft Teams application.
4. Select your resource group for the bot.
5. Choose your pricing plan.
6. For **Type of app**, choose the **Multi tenant** option.
7. For **Creation type**, choose **Use existing app registration**. This is how you link the bot to your Microsoft Teams application. Enter these values from the Teams application you created, as described in Creating a Microsoft Teams Application for IdentityIQ in Azure.
    - **App ID**
    - **App secret**
8. Click **Review + Create** to create the bot and link it to your Microsoft Teams application. On the review screen, click **Create** to confirm the bot creation.
9. When the deployment process is complete, configure the bot:
    a. Click **Go to Resource**.
    b. In the left navigation, click **Configuration**.
    c. Enter a **Messaging endpoint**. This endpoint is used by the Microsoft Teams application to communicate with the bot. The endpoint must be a public, DNS-resolvable domain name.

       For security purposes, SailPoint recommends using a format that incorporates the **Microsoft App ID**

with the dashes removed. For example:

`https://identityiqteams.mycompany.com:3978/appidwithnodashes/api/messages`

d. Click **Apply**.

e. Click **Add OAuth Connection Settings** – this connection manages tokens for SSO authentication. You will use some values from the Microsoft Teams application you created; see Creating a Microsoft Teams Application for IdentityIQ in Azure for details on where to find those values.

   i. Enter a **Name** for the new connection setting. Do not include spaces in the name.

   ii. For **Service provider**, select **Azure Active Directory v2**.

   iii. Enter the **Client ID** and **Client secret** from your Microsoft Teams application.

   iv. For **Token Exchange URL** enter the **Application ID URI** from your Microsoft Teams application. This is the URI that begins with `api://botid-`

   v. Enter your **Tenant ID**.

   vi. For **Scope**, enter the scope you created for your API application. To find this value, navigate to your API application, click **Expose an API**, and copy the full scope from the **Scopes** field there, to enter here. See Creating an API Application in Azure.

   vii. Click **Save**.

10. Click **Apply** to save all your changes.

Installing and Configuring the IdentityIQ Service Code

The IdentityIQ service code is provided by SailPoint. The service code is packaged into platform-specific zip files and is downloaded from Compass. Choose the zip file appropriate for your platform.

The Linux zip file includes an executable file, an `env.template` file, and a shell script and service script to assist with installing the code as a service.

The Windows zip file contains an executable file and an `env.template` file.

Once you have downloaded the zip appropriate to your environment, follow these steps:

1. Extract the zip to the private server dedicated to hosting the service code. See the section on Prerequisites for Integrating Microsoft Teams with IdentityIQ for more information.

2. The zip file includes a template for configuring your IdentityIQ/Microsoft Teams environment, named `env.template`. Copy this file and name it `.env`.

   > Note: Once the zip file has been extracted, add a `BOT_HOME` environment variable to the server, set to the directory where your executable file, environment file, and `cert` directory are located (see steps below for details on the `cert` directory).

3. Edit the `.env` file to set configuration values for your environment. The values you *must* configure are listed below; some other values can be modified as needed for your environment. The variables in the file are commented, to give you information on their usage.

   - `PUBLIC_HOSTNAME=<public DNS hostname>`: the public domain name of where you host the IdentityIQ service code. This must be a public DNS-resolvable hostname that resolves to the private IP of the server that will run the service code. The traffic can be limited to a specific port, which is customizable. See Prerequisites for Integrating Microsoft Teams with IdentityIQ.
   - `LOCAL_HOSTNAME=<local private IP address>`: this is the private IP to which the `PUBLIC_HOSTNAME` is routed.
   - `PUBLIC_PORT` and `PRIVATE_PORT`: the hosting ports for the IdentityIQ service code. The template provides default values which can be replaced with values specific to your installation. See the **Connectivity and Security Prerequisites** section in Prerequisites for Integrating Microsoft Teams with IdentityIQ.
   - `TENANT_ID=<Azure tenant ID>`: your Azure tenant ID.
   - `APP_ID=<teams app ID>`: the Application (client) ID value that was set as part of configuring the Microsoft Teams app. You can find this value on the Overview page of the Microsoft Teams app. See Creating a Microsoft Teams Application for IdentityIQ in Azure
   - `APP_NAME=<teams app name>`: the name of your Microsoft Teams app. See Creating a Microsoft Teams Application for IdentityIQ in Azure
   - `APP_SECRET=<encrypted teams app secret>`: **IMPORTANT**: this value is required but will be set later, in a secondary step. See step 5 below for more information.
   - `SSO_CONNECTION_NAME=<sso connection associated with azure bot>`: this value was set in the **Add OAuth Connection Settings** when you configured the Azure bot. See Creating an Azure Bot for IdentityIQ's Microsoft Teams Notifications.

- `ENCRYPTION_SECRET=<provide strong value>`: provide an 8-character encryption secret. This can be any value you like, but if this value changes, any items encrypted must be re-encrypted
- `IIQ_URL`: the full URL to your installation of IdentityIQ. Use this format:

  `https://<host/ip>:<port>/<identityiq_home>`

  where `<identityiq_home>` is the directory in which you extracted the identityiq.war file during the IdentityIQ installation procedure. If you are using a load balancer to manage multiple IdentityIQ hosts, put the load balancer URL here.

4. Install valid certificate files in the `cert` directory contained in the root location where your extracted service code and `.env` file are. The certificate must be named `bot.cert` and the key must be named `bot.key`.

   The service code can work on encrypted or raw certificates, but for security reasons, it is recommended to encrypt them. For convenience, an encrypt endpoint has been provided with the service code. Follow these guidelines for creating encrypted `bot.cert` and `bot.key`:

   - Convert the raw certificate and key to **single lines** by replacing end of line characters with \n
   - POST the single-line certificate and key to the encryption endpoint.
   - Paste the responses to the POST operation into new files called `bot.cert` and `bot.key` and save them. After creating the files, verify that an end of line character was not inserted by the editor; remove the end of line character if there is one.
   - Make sure the `cert` directory, the `bot.cert` and the `bot.key` items are set with `rw` permission for owner only.
   - For more information on certificates, see Prerequisites for Integrating Microsoft Teams with IdentityIQ.

5. Encrypt and add your client secret to the environment file. For convenience, the IdentityIQ service code provides an encryption endpoint.

   a. Make sure rest of the `.env` file has been configured and saved before encrypting and adding the client secret.

   b. Start the bot (for example, by running the `identityiqbot.sh` script). Note that because the secret has not yet been encrypted, **you may see an error when initially starting the bot.** You can continue past this error to complete the process for encrypting the secret.

c. Use the following private POST endpoint to encrypt messages such as bot secret, certificate, and private key. Provide a JSON object as the payload and a JSON object will be returned with the encrypted value.

```
https://<private IP>:<private port>/util/encrypt
```

```
Input: {"message" : "some message to be encrypted"}
```

```
Output: { "status" : "success", "encrypted_message" : "……."}
```

d. Copy the returned value into the `APP_SECRET=<encrypted teams app secret>` field of the `.env` file.

e. Restart the bot.

Allowing Microsoft Teams Users to Disable Notifications

The `ALLOW_USER_TO_DISABLE_NOTIFICATIONS` property in the `.env` file allows Microsoft Teams users to disable notifications, by entering the command *nonotifications* in their Microsoft Teams environment. This option is disabled by default; if you want to allow users to disable notifications, set this value to `true`.

Creating a Microsoft Teams Manifest

Once the `.env` file is completely configured, a bot manifest can be created. This is the file users will use to install the bot into their local Microsoft Teams environments.

To create a manifest, use the private `POST` endpoint, which is provided with the service code, to generate the manifest. This endpoint requires information from the `.env` file as input. When executed, the endpoint generates a manifest zip file in the `<BOT_HOME>` location.

The response also includes the contents of the `manifest.json` file (as included in manifest zip file).

```
https://<private IP>:<private port>/util/manifest
```

```
Input: {"tenant_id" : "xxxxxxxx", "app_id" : "yyyyyyy"}
```

```
Output: {"status" : "success", "manifest" : {……}}
```

After a successful post to the manifest endpoint, a zip file named `identityiqbot-<version>-manifest.zip`.is created in `<BOT_HOME>`.

The manifest zip file contains the `manifest.json` file, as well as some configuration options such as color and outline logos.

Once the manifest is generated, it doesn't need to be generated again unless there are changes to the bot configuration in the `.env` file – for example, due to Azure portal changes.

> If there is trouble generating a manifest, verify the tenant_id and app_id are correct.

For more information on the service code, see Installing and Configuring the IdentityIQ Service Code.

Configuring IdentityIQ's Microsoft Azure Active Directory Application for Teams

To enable IdentityIQ to send notifications to Microsoft Teams, you must add details for connecting to the Microsoft Teams Bot the Azure Active Directory application in IdentityIQ. The values you use here correspond to the configuration settings for your Microsoft Teams and bot applications in Azure.

This section assumes you have already created an Azure Active Directory application in IdentityIQ and registered it in Azure. For details on creating an Azure Active Directory application, refer to SailPoint's *Integrating SailPoint with Azure Active Directory Connector* guide, which is available in the **Microsoft Azure Active Directory** area of the Connector Directory on Compass.

To configure Teams-specific settings in the application:

1. Click **Applications > Application Definition**.
2. Open your **Azure Active Directory** application.

   > For complete details on creating and configuring an Azure Active Directory application, refer to the *Integrating SailPoint with Azure Active Directory Connector* guide.

3. Click **Configuration > Settings**.
4. Configure these values for Microsoft Teams in the **Additional Configuration** section:
   - **Enable Microsoft Teams Notifications** – check this option to enable IdentityIQ notifications within Microsoft Teams. Note that this option must be checked in order for the next fields to appear.
   - **Microsoft Teams Bot URL** – the private URL to the server hosting the IdentityIQ service code, including the port and endpoint, in the format `http://server:port/appidwithnodashes/api/notify` where *`appidwithnodashes`* is the Application (client) Id for the Microsoft Teams application. This ID was also used when you set up the **Messaging endpoint** in the Azure bot. Note that the step in Creating an Azure Bot for IdentityIQ's Microsoft Teams Notifications that include this **Messaging endpoint** references the *public* server; in this field in IdentityIQ, be sure to use server and port details for the *private* server.
   - **Tenant ID** – the Azure Active Directory tenant ID that is used for your Microsoft Teams application.

- **Microsoft Teams Bot Secret** – the client secret you generated when creating the Microsoft Teams application. See Creating a Microsoft Teams Application for IdentityIQ in Azure for details.

5. **Save** your changes.

Enabling Microsoft Teams Notifications Within IdentityIQ

You must globally enable Microsoft Teams within IdentityIQ, to allow IdentityIQ to send notifications to Microsoft Teams users.

1. Click on the **gear** icon and navigate to **Global Settings > IdentityIQ Configuration > Notifications Settings**.
2. Select **Enable Microsoft Teams Notifications**. This turns on notifications and enables your Microsoft Teams users to receive them.
3. **Notification Templates** determine the contents of the notifications that are sent in Microsoft Teams. You can choose only one notification template for each type of notification (for example, **For reminder notices**, **For escalation notices**, etc.).

> Important: IdentityIQ uses a single notification template for each type of notification; in other words, for a specific type of notification, you cannot specify one template for email notifications and a different template for Microsoft Teams notifications.

For information on how to edit Notification Templates, see the *SailPoint IdentityIQ System Configuration Guide*.

Formatting Options for Microsoft Teams Notifications

There is a property in the `.env` file that identifies the format of notification messages: `NOTIFICATION_TEXT_FORMAT`. The valid values are `xml`, `markdown`, and `plain`.

If the property has not been set in the `.env` file, or is not a valid value, it will default to `markdown`. For more details on how the text format is used by Teams, refer to Microsoft's documentation on formatting bot messages.

For more information on editing `.env` properties, see Installing and Configuring the IdentityIQ Service Code

Configuring API Authentication for Microsoft Teams in IdentityIQ

To enable token-based authentication between Microsoft Teams and IdentityIQ, you must create an API application in Azure, then use some of the values from the API application within IdentityIQ, to enable authentication. For related information about what to configure in Azure for token-based authentication, see Creating an API Application in Azure

To configure API Authentication in IdentityIQ:

1. Click **gear > Global Settings > API Authentication**.
2. On the **General Settings** tab, set an expiration value for tokens, in seconds. This value applies to both **Token Settings** (see below) and **OAuth Client Management** (see **API Authentication** in the **System Configuration** documentation).
3. Click the **Token Settings** tab, and set these values:
    - **Access Token Authentication Scope** – expected scope of the API access token issuer. This is the value you entered for **Scope Name** in the previous section; for example, `GetToken`.
    - **Access Token Authentication Audience** – suffix that identifies the service or system to which the call is directed. This is the value you appended to the **Application ID URI** in the Creating an API Application in Azure section; for example `/identityiq/api`. The validator will ensure the SSO audience claim ends with this value.
    - **Access Token Authentication Issuers** – identification of the SSO token provider. This field supports variable interpolation. Field values must be claims in the SSO token.
      For example, `https://sts.windows.net/{{tid}}/`.
    - **Correlation Variable** – the SSO claim used match the requesting user with an existing IdentityIQ user; the IdentityIQ default is `oid`.
4. **Save** your changes.

For more information on configuring your IdentityIQ instance for API authentication, refer to the **API Authentication** section of the *SailPoint IdentityIQ System Configuration Guide*.

Installing the IdentityIQ Microsoft Teams Notifications Bot in Microsoft Teams

Each Microsoft Teams user must install the IdentityIQ bot in their Microsoft Teams environment. The manifest you created in Creating a Microsoft Teams Manifest can be made available by your Microsoft Teams administrator to Microsoft Teams users in an approved apps list.

Microsoft Teams users simply follow the instructions for installing apps from the list. For more information, refer to Microsoft's documentation.

When the bot is installed, it sends a `notifications` command to IdentityIQ. This makes the connection between the user's Microsoft Teams environment and your installation of IdentityIQ

> Note: If the initial installation's `notifications` command does not result in an *"IdentityIQ notifications have been enabled"* response, wait a few minutes, then manually issue the

> `notifications` command. You will receive the expected response once Microsoft Teams is communicating with IdentityIQ.

> If at any time after Microsoft Teams Notifications has been implemented, a user is no longer receiving expected notifications from IdentityIQ, issue the `notifications` command from the Microsoft Teams console. An *"IdentityIQ notifications have been enabled"* response indicates that the connection has been restored.

Using IdentityIQ Microsoft Teams Notifications

Microsoft Teams users receive notifications from IdentityIQ directly in their Microsoft Teams client. Notifications include links to IdentityIQ, to take the user to the relevant action or area in IdentityIQ.

If single sign-on (SSO) between Microsoft Teams and IdentityIQ has been configured, links take Microsoft Teams users directly to the appropriate destination in IdentityIQ, without requiring a step for logging in to IdentityIQ. If SSO has not been enabled, users will need to log in to IdentityIQ.

Microsoft Teams users can use these commands:

- *hello* – returns a response with the version
- *help* – displays the bot help message
- *identityiq* – verifies the connection between IdentityIQ and Microsoft Teams
- *notifications* – makes the connection between the user's Microsoft Teams environment and their installation of IdentityIQ

Auditing Microsoft Teams Notifications

IdentityIQ's audit configuration includes two audit actions related to Microsoft Teams notifications, that can be collected for audit logs:

- Chat Notification Success
- Chat Notification Failure

For more information about auditing in IdentityIQ, see the *SailPoint IdentityIQ System Configuration Guide*.