
FIDO Pre-reg

Yubico

Sep 30, 2026

API REFERENCE AND RELEASE NOTES

1	Introduction	1
1.1	How it Works	1
1.2	API Version and Feature Support	2
1.3	Environment Variables	4
2	Multi-IdP Support	7
2.1	ARM Template Parameters	7
2.2	Enabling Multi-IdP Support	8
2.2.1	Configuration File Example	9
2.2.2	Option 1: Container Volume Mount	10
2.2.3	Option 2: Blob Storage Container	10
3	API Reference	13
3.1	Authentication	13
3.2	FIDO Connector API	13
3.2.1	Create Credential Request	13
3.2.1.1	Example 1: Credential-only Request	18
3.2.1.2	Example 2: Shipment-only Request	19
3.2.1.3	Example 3: Shipment and Credential Request	21
3.2.1.4	Example 4: Multi-IdP Credential-only Request	22
3.2.1.5	Version 1 - Create Credential Request	23
3.2.2	Get Credential Request by ID	24
3.2.2.1	Version 1 - Get Credential Request by ID	26
3.2.3	Check Deployment Status	26
3.2.3.1	Version 1 - Check Deployment Status	28
3.2.4	Process Shipments	28
3.2.5	Process Credential Request	29
3.2.5.1	Version 1 - Process Credential Requests	29
3.2.6	Resend PIN	29
3.2.6.1	Version 1 - Resend PIN	30
3.3	FIDO Connector API and FIDO Pre-reg	30
3.3.1	Create Shipment Request	30
3.3.2	Get Shipment Request Status	32
4	Release Notes	35
4.1	2026	35
4.1.1	Release 2.0.9 (29 September 2026)	35
4.1.2	Release 2.0.8 (17 September 2026)	35
4.1.3	Release 2.0.7 (16 September 2026)	35
4.1.4	Release 2.0.4 (30 July 2026)	36

4.1.5	Release 2.0.3 (21 July 2026)	36
4.1.6	Release 2.0.1 (15 June 2026)	36
4.1.7	Release 2.0.0 (15 June 2026)	37
4.1.8	Release 1.2.10 (15 June 2026)	37
4.1.9	Release 1.2.8 (6 May 2026)	38
4.1.10	Release 1.2.7 (16 April 2026)	38
4.1.11	Release 1.2.3 (23 February 2026)	39
4.2	2025	39
4.2.1	Release 1.2.0 (3 November 2025)	39
4.2.2	Release 1.1.6 (16 October 2025)	40
4.2.3	Release 1.1.4 (22 August 2025)	40
4.2.4	Release 1.1.1 (25 June 2025)	41
4.2.5	Release 1.0.0 (3 April 2025)	41
5	Copyright	43
5.1	Trademarks	43
5.2	Disclaimer	43
5.3	Contact Information	43
5.4	Getting Help	44
5.5	Feedback	44
5.6	Document Updated	44

INTRODUCTION

The FIDO Connector and associated APIs provided by Yubico with [YubiKey as a Service - Enrollment](#), are used for FIDO Pre-reg integrations with identity providers (IdPs) to manage system interactions supporting pre-enrollment of credentials and shipment of YubiKeys.

Note: The Microsoft Entra ID integration with FIDO Pre-reg and the FIDO Connector is currently available in *Early Access (EA)*. The Enroll app is currently available in *Limited Early Access (LEA)*.

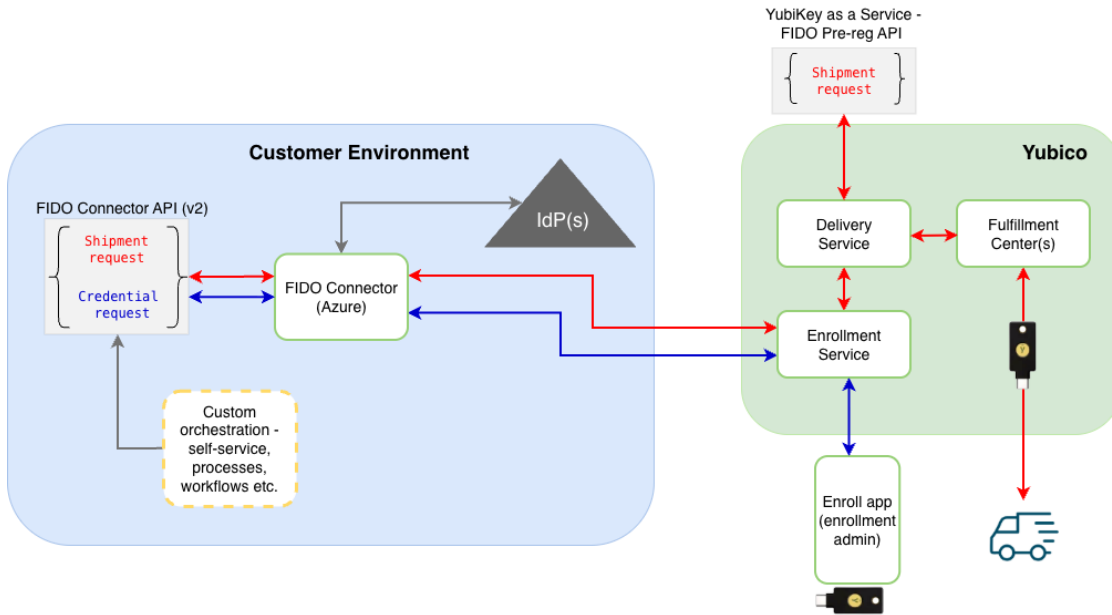
1.1 How it Works

The FIDO Connector is an orchestration component used in enrollment to handle credential and shipment requests. The Connector is deployed in the customer's Azure environment, and provides APIs for for credential request creation and shipment of enrolled YubiKeys.

The current version of the FIDO Connector API supports the following features:

- **Multiple identity providers:** Create multiple credential requests to one or more IdPs, useful if your organization has more than one IdP for system authentication.
- **Multiple credentials:** Add multiple credentials on the same YubiKey, for example for different account types, as required by your end users.
- **Multiple keys:** Include, for example, a primary and a backup YubiKey programmed for the same account credentials, in a single request.
- **On-site programming:** Use the [Enroll App \(LEA\)](#) on a mobile device to program a YubiKey with credentials at your premises before it is handed over to an end user.

The image below provides an overview of the architecture with the FIDO Connector, one or more IdPs, the integration APIs used, and the services provided by Yubico.



The following components are included:

- **Identity providers (IdPs):** The FIDO Connector supports Microsoft Entra ID, PingOne PingID, and PingOne AIC, see [FIDO Pre-reg integration guides](#).
- **Customer environment:** IT system components, including the Azure environment where the FIDO Connector is deployed, see for example [FIDO Pre-reg with Microsoft](#).
- **Business system integrations:** Integrations with for example an HR system providing recipient address information, or ServiceNow for triggering shipments of physical keys.
- **Delivery Service:** The Yubico-provided APIs for managing shipments of physical keys, with access to the Yubico Customer Portal for managing shipment statuses.
- **Enrollment Service:** Manages credential and shipment request data, provides APIs for managing on-site programming requests.
- **Fulfillment Center(s):** Yubico-managed programming and shipping of physical YubiKeys globally.
- **Enroll app:** Yubico-provided mobile app that lets an authenticated user (enrollment administrator) process programming of credentials, locally at their premises.
- **API Calls:** Calls to the [FIDO Connector](#) and [YubiKey as a Service-FIDO Pre-reg](#) APIs for requesting credentials and/or shipments of enrolled YubiKeys.

1.2 API Version and Feature Support

The following FIDO Pre-reg integration options are available:

- Using the [FIDO Connector API v2 and v1](#)
- Using the [FIDO Pre-reg API \(fido2PreRegisteredShipments\)](#)

The table below provides an overview of supported features for each integration option.

Feature	FIDO Connector v2	FIDO Connector v1	FIDO Pre-reg
Support for IdPs Microsoft Entra ID, PingOne PingID, and PingOne AIC (see note below).	Yes	Yes	Yes (see note below)
Credential requests to multiple IdPs in a single request.	Yes	No	Yes (see note below)
Multiple credentials on the same key in a single request.	Yes	Yes	No
Multiple keys with multiple credentials in a single request.	Yes	No	Yes (see note below)
On-site credential programming of keys for handover to an end user locally.	Yes	Yes	No
End-user provided PIN when programming keys locally on-site.	Yes	Yes	No
Shipment requests and credential requests in the same request.	Yes	No	No
Shipment-only request of one or more keys, programmed and shipped by Yubico.	Yes	Yes (one key only)	Yes (see note below)

Note: The Okta integration only supports the FIDO Pre-reg API, not the FIDO Connector API, and only one enrolled key at the time can be shipped. See the [FIDO Pre-reg with Okta Integration Guide](#).

1.3 Environment Variables

The following table lists environment variables that are common for FIDO Pre-reg integrations with different IdPs. The configured values can be updated in the Azure Container when needed. To obtain the status of current deployment components, see [Check Deployment Status](#).

Note: A manual restart of the Azure Container might be needed for the updates to the environment variables to be recognized.

Variable	Default value (if not changed by Environment Variable)	Comment
CRON_PROCESS_SHIPMENT_SCI	0 0 * * * *	The cron expression for the schedule that checks ongoing shipments and processes them (v1 API). The default setting is to run every hour at the top of the hour. It can be changed to any valid cron expression, for example to run every 30 minutes: 0 */30 * * * *
CRON_PROCESS_CREDENTIAL_I _SCHEDULE	0 0 * * * *	The cron expression for the schedule that checks ongoing credential/shipment requests and processes them (v2 API). The default setting is to run every hour at the top of the hour. It can be changed to any valid cron expression, for example to run every 30 minutes: 0 */30 * * * *
LOGGING_LEVEL_COM_YUBICC	INFO	Controls the level of information that is logged by the application. Can be increased to DEBUG if trying to solve a configuration issue. For normal usage, it is highly recommended to leave it as INFO.
ENTRA_FIDO_API_CHALLENGE_NUTES	20160	Numeric value representing time in minutes that the Entra FIDO challenge is valid for.

1.3. Environment Variables

5

CRON_DATA_CLEANUP_COMPLI 0

The below cron schedule will purge 'complete' shipment records that

Note: The encrypted credential response including the PIN, is automatically purged after 90 days. Data associated with the shipment request itself, for example recipient name and address, email address, and telephone number, is kept by Yubico for 7 years, in order to comply with export control and financial legal requirements.

MULTI-IDP SUPPORT

The multi-IdP feature enables the FIDO Connector to operate with multiple IdP instances. The FIDO Connector (v2) can be configured to support one or more of the following IdPs:

- Microsoft Entra ID
- PingOne PingID
- PingOne AIC

For information about these integrations, see [FIDO Pre-reg with Microsoft](#) and [FIDO Pre-reg with PingOne](#).

2.1 ARM Template Parameters

The ARM template provided by Yubico contains the following primary parameters for enabling or disabling multi-IdP support for the FIDO Connector:

- **ShouldUseMultiIdp**: Determines whether the FIDO Connector operates in single-IdP mode, configured via environment variables only, or multi-IdP mode where the configuration is pulled from an external file (`multi-idp-config.yaml`).

The parameter has the following values, available for selection in a drop-down menu when deploying the ARM template: “No”, “Yes, configured via Container Volume Mount file”, “Yes, configured via Blob Storage file”.

- **defaultIdentityProvider**: Defines the primary identity provider used by the FIDO Connector. The following values are available for selection when deploying the ARM template: “EntraID”, “PingOne PingID”, or “PingOne Advanced Cloud Identity”. The behavior depends on whether multi-IdP is disabled or enabled:
 - Multi-IdP is *disabled* (`ShouldUseMultiIdp = “No”`): Controls which single IdP to be used. The ARM template will automatically inject the corresponding environment variables, for example client IDs and endpoints, required for that specific IdP.
 - Multi-IdP is *enabled*: A configuration file is used to support multiple IdPs. In this mode, `defaultIdentityProvider` acts as a fallback, telling the FIDO Connector which IdP should be the default when nothing is specified in the requests.

The selected configuration method when enabling multi-IdP support will impact the deployed resources and the required post-deployment steps. For more information, see [Option 1: Container Volume Mount](#) and [Option 2: Blob Storage Container](#).

The `ShouldUseMultiIdp` and `defaultIdentityProvider` parameter settings impact the following:

- **FIDO Connector environment variables**:
 - The *single-IdP* mode injects individual environment variables, for example `ENTRA_FIDO_API_CLIENT_ID` or `PINGONE_CLIENT_ID`.

- The *multi-IdP* mode bypasses these variables, setting `IDP_ENABLE_MULTI_IDP` to “true” and defines the file lookup path via `IDP_CONFIGURATION_PATH`.

- **Storage and container architecture:** The *Volume mount* method creates container secrets and disk mounts. The *Blob storage* method provisions Blob components and RBAC roles.

Regardless of the single-IdP/multi-IdP settings, the ARM template always deploys the following:

- Virtual Network (VNet), subnets, and private endpoints.
- Azure Storage Account, including Azure Table Storage for shipment tracking.
- Azure Key Vault secrets for all supported IdPs (Entra ID, PingOne PingID/AIC).
- Container App instance, Log Analytics Workspace, App Insights, and Logic App workflows.

2.2 Enabling Multi-IdP Support

Multi-IdP support is disabled by default, and must first be enabled in your FIDO Connector instance. The *parameters in the ARM template* lets you enable multi-IdP support, and choose the method for providing the IdP-specific configurations. You can provide these either by using the *Volume mount method*, or the *Blob storage container method*.

Note: Version 1 of the FIDO Connector does not support multiple IdPs, the `IDP_DEFAULT` that has been configured will always be used. If you are migrating from v1 to v2 of the FIDO Connector, and want to enable multi-IdP and add a new IdP, consider configuring your existing IdP as `default_idp` for processing existing requests, and use the new IdP for new requests.

To enable multi-IdP support in the FIDO Connector in your Azure environment, do the following:

1. Create a configuration YAML file containing the settings for your IdPs. Each IdP instance must have a unique identifier, see *Configuration File Example*.
2. Provide the IdP-specific configurations using *one of the following options*:
 - Update the configurations in the placeholder file in the Container App Secret, see *Option 1: Container Volume Mount*.
 - Upload the configuration file to a Blob container, see *Option 2: Blob Storage Container*.
3. Configure Environment variables in the Container App as follows:
 - **IDP_ENABLE_MULTI_IDP:** Set to “true”.
 - **IDP_DEFAULT:** Set to your primary IdP’s unique identifier. Note that when using multiple IdPs, `IDP_DEFAULT` must be one of the identifiers used in the configuration file created in step 1.
 - **IDP_CONFIGURATION_PATH:** Configuration YAML file name and path, derived from either of the options in step 2. Example: “/app/config/idp-configuration.yaml”.
 - **AZURE_STORAGE_BLOB_CONTAINER_NAME:** If using Blob Storage, also provide the Container name of the Blob Storage.
 - **AZURE_STORAGE_BLOB_ENDPOINT:** If using Blob Storage, also provide your Azure endpoint. Example “https://msfprstorage.blob.core.windows.net/”.

In your implementation you can use multiple instances of the same IdP, for example, two PingOne PingID instances. Or you can have multiple instances of all supported IdPs. The `credentials.idp_id` in the request provides information about which IdP to use for each set of credentials. If this is not provided, `IDP_DEFAULT` will be used.

2.2.1 Configuration File Example

The following shows an example of a multi-IdP YAML configuration file using one instance each of Microsoft Entra ID, PingOne PingID, and PingOne AIC.

```
idp:
instances:
  entraid: # Unique random name of the instance (for example "entraid-123"), only
↳ letters and numbers.
    type: entraid
    secrets: # More than one secret can be used
      client-secret-id: STRING_VALUE # Use keys from KeyVault
    online-endpoint: STRING_VALUE
    graph-endpoint: STRING_VALUE
    challenge-timeout-minutes: NUMBER_VALUE
    api-version: STRING_VALUE
    client-id: STRING_VALUE
    tenant-id: STRING_VALUE
  pingone:
    type: pingone
    environment-id: STRING_VALUE
    client-id: STRING_VALUE
    secrets:
      client-secret-id: STRING_VALUE
    auth-base-url: STRING_VALUE
    api-base-url: STRING_VALUE
    pre-registration-timeout-days: NUMBER_VALUE
    default-relying-party: STRING_VALUE
  ping-aic:
    type: ping-aic
    realm: STRING_VALUE
    auth-base-url: STRING_VALUE
    client-id: STRING_VALUE
    api-base-url: STRING_VALUE
    scope: STRING_VALUE
    journey: STRING_VALUE
    authentication-header: STRING_VALUE
    callback-1: STRING_VALUE
    callback-2: STRING_VALUE
    callback-3: STRING_VALUE
    token-id-1: STRING_VALUE
    token-id-2: STRING_VALUE
    challenge-timeout-milliseconds: NUMBER_VALUE
    default-relying-party: STRING_VALUE
    secrets:
      client-secret-id: STRING_VALUE
```

2.2.2 Option 1: Container Volume Mount

The following describes how provide the IdP-specific configurations for multi-IdP support, using the Container volume mount method, configured through “ShouldUseMultiIdp = Yes, configured via Container Volume Mount file”.

The content of the configuration file is stored in the Container Secret’s value. The secret is then used to create a Container Volume. This has a *file path* field which will become the file name, for example “idp-configuration.yaml” used in the multi-IdP configuration.

The Volume Mount has a field for *mount path*, the directory where the Volume is placed, for example “/app/config”. The IDP_CONFIGURATION_PATH environment variable must then be pointed to the Volume Mount’s mount path + Volume’s file path. Example: “/app/config/idp-configuration.yaml”.

- **What happens in Azure:** The ARM template automatically creates an Azure Container App secret containing a boilerplate multi-IdP YAML structure (exampleMultiIdpConfigYaml). It then mounts this secret directly into the container as a file at /app/config/multi-idp-config.yaml.
- **Post-deployment step:** To update the placeholder secret generated by the ARM template, you only need to edit the placeholder values directly within the Container App’s secrets management.

If you are *not using the ARM Template*, and want to change the filename or directory (not needed to make the application start), then follow the procedure described below.

To mount and store a volume file from scratch, do the following:

1. Sign in to the [Azure Portal](#), go to your **Container App > Security > Secrets**.
2. Click **Add**, enter your Secret Key, select **Container App Secret** and, in value, enter your configuration YAML file content. Click **Save**.
3. Update the content of the Container App Secret as follows:
 - a. In the **Container App** screen, go to **Volumes**, then click **Add**.
 - b. Add your volume name.
 - c. Deselect **Mount all secrets**, click the dropdown and select the secret you created in the previous step, and as **File Path**, add the YAML file name. Example: “idp-configuration.yaml”.
 - d. In the **Container App** screen, go to **Containers** and select the **Volume mounts** tab.
 - e. Click **Add**, select your volume in the **Volume name** dropdown, then add your **Mount path**. The path should start from the /app directory, for example /app/config.
 - f. Still in **Containers**, go to the **Environment variables** tab, and add a variable IDP_CONFIGURATION_PATH=config/{FILE_PATH} Example: IDP_CONFIGURATION_PATH: config/idp-configuration.yaml.
4. Configure the Environment variables as described in [Enabling Multi-IdP Support](#).

2.2.3 Option 2: Blob Storage Container

The following describes how provide the IdP-specific configurations for multi-IdP support, using the Blob storage container method, configured through the setting “ShouldUseMultiIdp = Yes, configured via Blob Storage file”.

- **What happens in Azure:** The ARM template provisions an Azure Storage Blob Service, creates a private Blob container named fidoconnectorblobs, and grants the Container App permission (*Storage Blob Data Contributor*) to read from it.

- **Post-deployment step:** ARM/Bicep templates cannot natively generate file contents inside a Blob container during infrastructure provisioning. Therefore you must manually upload your multi-idp-config.yaml file to the newly created `fidoconnectorblobs` container in order for the FIDO Connector to successfully ingest multiple provider configurations.

To add the file to a Blob storage, do the following in the the [Azure Portal](#):

1. Create a Blob storage account:

- In the search bar at the top, type “Storage accounts” and select your account.
- Click **+ Create**.
- On the **Basics** tab, enter the following:
 - **Subscription:** Select your active subscription.
 - **Resource Group:** Select your existing Resource Group.
 - **Storage account name:** Enter a unique name, for example “yubifprstorage”, use only lowercase letters and numbers.
 - **Region:** Choose the same region as your Resource Group.
 - **Performance:** Standard is recommended for most use cases.
 - **Redundancy:** Select the desired redundancy, for example “Locally-redundant storage (LRS)”.
- Click **Review + create** and then **Create**. Wait for the deployment to finish.

2. Create a Container and upload the configuration file:

- If you are using the [ARM Template Parameters](#) (FIDO Connector v2) when configuring multi-IdP, continue to step 2b to upload the configuration file.

If you are *not* using the ARM template, do the following:

- Once the storage account is created, go to the new **Storage account** resource.
- In the left menu, under **Data storage**, click **Containers**.
- Click **+ Container** and enter the following:
 - **Name:** Enter “fidoconnectorblobs”, or the name expected by your application.
 - **Public access level:** Use “Private” (no anonymous access).
- Click **Create**.
- Click on the newly created container (“fidoconnectorblobs”).
- Click **Upload**.
- In the **Upload blob** panel, select your configuration file, for example “idp-configuration.yaml”.
- Click **Upload**.

3. Assign “Storage Blob Data Contributor” Role to Container App:

- Go to your **Storage account**.
- In the left menu, click **Access Control (IAM)**.
- Click **+ Add** and select **Add role assignment**.
- On the **Role** tab, search for **Storage Blob Data Contributor**, select it and click **Next**.
- On the **Members** tab do the following:
 - Assign access to **Managed identity**.

- Click + **Select members**.
 - In the **Managed identity** dropdown, select **Container App**.
 - Find and select your **Container App** from the list.
 - Click **Select**.
- f. Click **Review + assign** to complete the process.
4. Configure the Environment variables as described in *Enabling Multi-IdP Support*

API REFERENCE

The following describes the APIs used with FIDO Pre-reg integrations when requesting credentials and shipment of enrolled YubiKeys.

3.1 Authentication

To authenticate to the YubiKey as a Service API and the FIDO Connector API, you will use an API caller authentication token (YED_API_Token). To create the API token, sign in to the Customer Portal with the account for the application that will be calling the APIs. See [API Caller Account Setup](#).

3.2 FIDO Connector API

The FIDO Connector is deployed in the customer's Azure environment, and provides APIs for credential requests. The API is called from processes and workflows to handle orchestration of credential requests from an IdP and shipment requests from Yubico.

Each deployment of the FIDO Connector will have its own instance of the API. The base URL is derived from the URL provided by the Container app service, and will be unique for each deployment. API calls will go to something like in the example below.

Base URL (example): `https://company-fido-connector.azurecontainerapps.io/...`

3.2.1 Create Credential Request

VERSION 2.0 (Current)

Provides the ability to create a request for both Yubico-managed and local on-site enrollment administrators programming of credentials onto a YubiKey using a mobile device. If desired, the request can include shipment requests for one or more enrolled YubiKeys. You can add multiple credentials to the same YubiKey, and request enrollment to multiple IdPs.

The *authenticator* in this context is a security application or device, for example a YubiKey. A *credential* is the authentication private/public key pair. One authenticator can have multiple credentials, and each credential correlates to a user ID and relying-party ID pair. A *credential request* is a request for creating a credential in the Yubico Enrollment service.

In the FIDO Connector API, all requests are credential requests, where some might also include shipments of physical YubiKeys. This means that there will always be a credential ID, and if shipment is included, there will also be a shipment ID. `shipment_items` is used for Yubico-managed programmed FIDO Pre-reg shipments, while `authenticators` is

used for local on-site programming of credentials onto a YubiKey without involving any shipping. Both shipment and authenticators can be included in a request.

When multiple credentials are registered on the same security key, the credential requests are processed in the same order as added in the `authenticators.credentials` array, and the responses are sent back in the same order. To control the order of the credentials used when signing in, for example in cases where the most recently registered credential is automatically used first, ensure to place the requests in the desired order in the `authenticators.credentials` array.

The endpoint receives the credentials, the authenticators, and if there are physical YubiKeys to be shipped, associated information for these. A recipient email address is required if the request involves a YubiKey shipment. The credential object contains a list of the names of the authenticators and/or YubiKeys to which the credentials will be added.

The FIDO Connector accepts any PIN length between 4 and 63 characters (inclusive). A 6-digit PIN is often used by mobile operating systems to increase authentication security, while remaining manageable to memorize by end users. Note that if the YubiKey is for example a YubiKey FIPS and requires 8 characters, the programming will set the `length` to 8.

Endpoint: POST `/v2/credential-requests`

Request body

```
{
  "recipient": {
    "email": "firstname.lastname@example.com",
    "first_name": "Firstname",
    "last_name": "Lastname",
    "company": "Example Corporation",
    "telephone": "+123456789",
    "external_id": "firstname.lastname@example.com"
  },
  "enrollment_contacts": {
    "emails_to": [
      "firstname.lastname@example.com", "itadmin@example.com"
    ]
  },
  "shipment": {
    "delivery_type": 1,
    "reseller_organization_id": "100",
    "customer_reference": "Cost center 123",
    "address_validation_bypass": false,
    "mailing_address": {
      "street_line1": "#123 Street Name",
      "street_line2": "Unit #456",
      "postal_code": "98765",
      "city": "City Name",
      "region": "MA",
      "country_code": "US"
    }
  },
  "shipment_items": [
    {
      "name": "My YubiKey",
      "customization_id": "ABC123",
      "product_id": "1",
      "inventory_product_id": "789",
      "pin_request": {
```

(continues on next page)

(continued from previous page)

```

        "length": 6,
        "type": "generate"
    },
    "credentials": [
        {
            "idp_id": "entraid",
            "user_id": "firstname.lastname@example.com"
        },
        {
            "idp_id": "pingone",
            "relying_party_id": "pingone",
            "user_id": "firstname.lastname@example.com"
        }
    ]
}
],
},
"authenticators": [
{
    "name": "My passkey",
    "customization_id": "DEF456",
    "preferred_authenticator_type": "software",
    "pin_request": {
        "min_length": 6,
        "entry_method": "user_entered",
        "delivery_method": "app"
    },
},
"credentials": [
    {
        "idp_id": "entraid",
        "user_id": "firstname.lastname@example.com"
    },
    {
        "idp_id": "pingone",
        "relying_party_id": "pingone",
        "user_id": "firstname.lastname@example.com"
    }
]
}
]
}
}

```

Parameters

- **recipient** The end-user receiving the pre-enrolled YubiKey and/or credentials:
 - **email** Required. Used for Yubikey as a Service emails such as delivery notifications for shipment of physical keys.
 - **first_name** Required.
 - **last_name** Required.
 - **company** Optional. Either the company name or recipient name is required, you can provide both if shipping to a commercial/business address.

- **telephone** Required. Supported characters are +, (,), and digits 0 - 9. Dashes are accepted but not recommended. Up to 40 characters allowed. Passed on to the delivery carrier to be used if there is an issue with delivering a package. Yubico does not use this field to contact the user.
- **external_id** Optional. Unique identifier for the recipient for which the credentials are requested, for example defined by the organization's HR system. Can be for example the end user's email address. If not provided, `recipient.email` is used instead to represent the user identity.
- **enrollment_contacts** A list of email addresses to, for example the end user or the security administrator, where the PIN is sent:
 - **emails_to** Required. List of valid email addresses, should at least include one email address, does not have to be the same as the recipient email.
- **shipment** Required if **authenticators** is not provided (shipment-only request), otherwise optional:
 - **delivery_type** Optional, if provided should be either "1" - standard (default), or "2" - expedited.
 - **reseller_organization_id** Optional, the organization ID of the channel partner (distributor/reseller) in the Delivery service. Use the [Get all channel partners relevant to an org](#) API to retrieve this.
 - **customer_reference** Optional, customer-provided reference information, see [Customer Reference Field](#).
 - **address_validation_bypass** Optional, "true/false", see [Address Validation Override](#).
 - **mailing_address** For shipment of physical keys:
 - * **street_line1** Required, first line of address, see [Address Field Limitations](#).
 - * **street_line2** Optional, second line of address. Can be used for long addresses, or to capture door code or other building access information, see [Address Field Limitations](#).
 - * **postal_code** Required, cannot be empty, space is accepted. For shipments to countries without postal codes, enter "00000".
 - * **city** Required.
 - * **region** Optional, required for shipments going to the US, Canada, and US international military bases. Two-letter USPS region code, see [USPS Region/State Codes](#).
 - * **country_code** Required. Two-letter country code, see [Country Codes](#).
 - **shipment_items** Used for shipment of Yubico-programmed YubiKeys:
 - * **name** Optionally provides the ability to add a custom passkey name in a pre-registration request. The field will be displayed together with the serial number of the YubiKey. For example, if the input name is "Secondary" and the serial number is "1234567890", this displays as "Secondary: 1234567890".

If a custom name is provided, all characters must be ASCII and the field length must not exceed 18 characters. If a custom name is not provided, the default name "Pre-Reg YubiKey: {SERIAL}" is displayed.
 - * **customization_id** Optional, ID associated with the programming, see [Customizations](#).
 - * **product_id** Required, ID of the YubiKey product model, see [Product and Inventory Identifiers](#).
 - * **inventory_product_id** Required, the subscription ID associated with the delivery, see [Product and Inventory Identifiers](#).
 - * **pin_request** Optional, defines the YubiKey PIN requirements. Default values only apply if at least the object is provided. Otherwise, the request is sent without the PIN request data. If an incomplete object is provided, the default **length** value is "6".
 - **length** Optional, if provided, should be equal to or greater than "4" and equal to or less than "63". The length of the PIN to be randomly generated and set on a YubiKey. Note this is not the

YubiKey firmware setting of minimum PIN length. If a length is requested that is less than the YubiKey setting of minimum PIN length, a PIN meeting that requirement will be generated.

- type Only “generate” is supported for shipments of physical keys. If not provided, the value defaults to “generate”.
- * credentials Required, information associated with the credentials:
 - idp_id Optional, used for multi-IdP deployments. Default value is configurable through the IDP_DEFAULT environment variable, see [identity provider integration guides](#).
 - relying_party_id Optional, default value can be configured in the IdP implementation. Used for example for PingOne, see [configuring PingOne PingID](#).
 - user_id ID of the end user in the relying party that the credential is being created for. Examples: For Entra ID, the UPN or user object ID; for PingOne, the PingOne PingID username.
- authenticators Optional, required if shipment is not provided (credential-only request):
 - name Optional authenticator “nickname”. If not provided, the default “Pre-Reg YubiKey” will be used..
 - customization_id Optional, ID associated with the programming, see [Customizations](#).
 - preferred_authenticator_type Optional, if provided can be either “hardware”, passkey on a physical key (default), or “software”, passkey on for example a mobile device.
 - pin_request Optional, defines whether a PIN is required or not. Default values only apply if at least the object is provided. Otherwise, the request is sent without the PIN request data. If an incomplete object is provided, the default value is “6”:
 - * min_length Optional, if provided, should be equal to or greater than “4” and equal or less than “63”.
 - * entry_method Optional, can be either “generated” (default), pre-programmed, or “user_entered”, entered during on-site enrollment.
 - * delivery_method Optional, if provided should be either “none”, “app”, or “email”.
 - credentials Required, information associated with the credentials:
 - * idp_id Optional, used for multi-IdP deployments. Default value is configurable through the IDP_DEFAULT environment variable, see [identity provider integration guides](#).
 - * relying_party_id Optional, default value can be configured in the IdP implementation. Used for example for PingOne, see [configuring PingOne PingID](#).
 - * user_id ID of the end user in the relying party that the credential is being created for.

Success response body

```
{
  "data": {
    "credential_request_id": "123abc-...",
    "shipment_id": "123xyz..."
  }
}
```

- data.credential_request_id The ID of the created credential request.
- data.shipment_id The ID of the created shipment request. Only included if shipment is a request parameter.

Error response body

```
{
  "error_code": "error type",
  "error_message": "error message",
}
```

(continues on next page)

(continued from previous page)

```
"error_data": {
  "key": "value"
}
}
```

Response messages

- 201 Created
- 400 Validation Failed or Propagated
- 401 Unauthorized on FIDO Connector
- 500 Unexpected error or failure authenticating with Enrollment service

3.2.1.1 Example 1: Credential-only Request

The following shows a request in a scenario where an enrollment administrator programs a YubiKey on-site, on behalf of an end user present in person. The end user is registered in the organization's IdP ("user_id"). The enrollment administrator already has the YubiKey, so no shipment is involved.

name provides the ability to add a custom passkey name in a pre-registration request. The field will be displayed together with the serial number of the YubiKey. In this example the input name is "Replacement", and if the serial number is "1234567890", this displays as "Replacement: 1234567890".

The request is received by the [Enroll app \(LEA\)](#), and the enrollment administrator uses the app on a mobile device to program the credentials onto the key ("preferred_authenticator_type": "hardware"). During the enrollment, the end user enters a PIN of their choice ("entry_method": "user_entered") in the mobile device. The enrolled YubiKey is then handed over to the end user. The request body contains an `authenticators` object with one credential item.

```
{
  "recipient": {
    "email": "enduser@pingone.com",
    "first_name": "Firstname",
    "last_name": "Lastname",
    "company": "Example Corporation",
    "telephone": "+46123456789",
    "external_id": "enduser@example.com"
  },
  "enrollment_contacts": {
    "emails_to": [
      "enduser@pexample.com, enrollmentadmin@example.com"
    ]
  },
  "authenticators": [
    {
      "name": "Replacement",
      "customization_id": "ABC123",
      "preferred_authenticator_type": "hardware",
      "pin_request": {
        "min_length": 6,
        "entry_method": "user_entered",
        "delivery_method": "app"
      }
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    },
    "credentials": [
      {
        "user_id": "enduser@example.com"
      }
    ]
  }
]
}

```

A temporary PIN can also be dynamically generated ("entry_method": "generated"), and emailed ("delivery_method": "email") to the enrollment administrator ("enrollment_contacts.emails_to"), as in the following example:

```

"enrollment_contacts": {
  "emails_to": [
    "enduser@example.com",
    "enrollmentadmin@example.com"
  ]
},
...
"pin_request": {
  "min_length": 6,
  "entry_method": "generated",
  "delivery_method": "email"
},

```

3.2.1.2 Example 2: Shipment-only Request

In this example scenario, two enrolled YubiKeys are shipped directly to a new employee working remotely. The request includes shipment of a YubiKey 5 NFC ("product_id": "1"), and a YubiKey 5C NFC ("product_id": "29"), to the end user who is registered in the organizations IdP ("user_id") Entra ID.

The keys are programmed by Yubico and shipped to the end user's home address. The 6-digit PIN ("length": 6) for each YubiKey is delivered through email to the end user. The request body contains a shipment object with two items.

```

{
  "recipient": {
    "email": "enduser@example.com",
    "first_name": "Firstname",
    "last_name": "Lastname",
    "company": "Example Corporation",
    "telephone": "+123456789",
    "external_id": "enduser@example.com"
  },
  "enrollment_contacts": {
    "emails_to": [
      "enduser@example.com"
    ]
  },
  "shipment": {

```

(continues on next page)

(continued from previous page)

```
"delivery_type": 1,
"customer_reference": "Cost center 123",
"address_validation_bypass": false,
"mailing_address": {
  "street_line1": "#123 Street Name",
  "street_line2": "Unit #456",
  "postal_code": "98765",
  "city": "City Name",
  "region": "MA",
  "country_code": "US"
},
"shipment_items": [
  {
    "name": "Primary YubiKey",
    "customization_id": "ABC123",
    "product_id": "1",
    "inventory_product_id": "133",
    "pin_request": {
      "length": 6,
      "type": "generate"
    },
    "credentials": [
      {
        "user_id": "enduser@example.com"
      }
    ]
  },
  {
    "name": "Backup YubiKey",
    "customization_id": "ABC123",
    "product_id": "29",
    "inventory_product_id": "133",
    "pin_request": {
      "length": 6,
      "type": "generate"
    },
    "credentials": [
      {
        "user_id": "enduser@example.com"
      }
    ]
  }
]
}
```

3.2.1.3 Example 3: Shipment and Credential Request

The following is an example of a scenario where one enrolled YubiKey is shipped to the end user as a backup key, and another YubiKey is programmed on site by an enrollment administrator, using the [Enroll app \(LEA\)](#) on a mobile device, with the end user present in person.

The 6-digit PIN for the YubiKey that is enrolled on site is entered by the end user during the enrollment procedure ("entry_method": "user_entered"). The shipped YubiKey is programmed by Yubico ("entry_method": "generated"), and the PIN is provided via email to the end user and the security administrator ("enrollment_contacts.emails_to"). The request body contains a shipment and an authenticators object with one item in each.

```
{
  "recipient": {
    "email": "enduser@example.com",
    "first_name": "Firstname",
    "last_name": "Lastname",
    "company": "Example Corporation",
    "telephone": "+123456789",
    "external_id": "enduser@example.com"
  },
  "enrollment_contacts": {
    "emails_to": ["enduser@example.com", "enrollmentadmin@example.com"]
  },
  "shipment": {
    "delivery_type": 1,
    "customer_reference": "Cost center 123",
    "address_validation_bypass": false,
    "mailing_address": {
      "street_line1": "#123 Street Name",
      "street_line2": "Unit #456",
      "postal_code": "98765",
      "city": "City Name",
      "region": "MA",
      "country_code": "US"
    },
    "shipment_items": [
      {
        "name": "Backup YubiKey",
        "customization_id": "ABC123",
        "product_id": "29",
        "inventory_product_id": "133",
        "pin_request": {
          "length": 6,
          "type": "generate"
        },
        "credentials": [
          {
            "user_id": "enduser@example.com"
          }
        ]
      }
    ]
  }
}
```

(continues on next page)

(continued from previous page)

```

},
"authenticators": [
  {
    "name": "Primary YubiKey",
    "customization_id": "ABC123",
    "preferred_authenticator_type": "hardware",
    "pin_request": {
      "min_length": 6,
      "entry_method": "user_entered",
      "delivery_method": "app"
    },
    "credentials": [
      {
        "user_id": "enduser@example.com"
      }
    ]
  }
]
}

```

3.2.1.4 Example 4: Multi-IdP Credential-only Request

In this example, an enrollment administrator for an organization that uses both IdPs Entra ID and PingOne, enrolls two credentials for an end user onto the same YubiKey, using the [Enroll app \(LEA\)](#) on a mobile device. After the programming, the enrollment administrator hands over the YubiKey to the end user, so no shipment is involved here.

The credentials are programmed onto the YubiKey ("preferred_authenticator_type": "hardware"). The temporary PIN is dynamically generated ("entry_method": "generated"), and then emailed ("delivery_method": "email") to the enrollment administrator ("enrollment_contacts.emails_to"). The request body contains an authenticators object with two credential items, one for each IdP.

```

{
  "recipient": {
    "email": "enduser@example.com",
    "first_name": "Firstname",
    "last_name": "Lastname",
    "company": "Example Corporation",
    "telephone": "+46123456789",
    "external_id": "enduser@example.com"
  },
  "enrollment_contacts": {
    "emails_to": ["enduser@example.com", "enrollmentadmin@example.com"]
  },
  "authenticators": [
    {
      "name": "System XYZ security key",
      "customization_id": "ABC123",
      "preferred_authenticator_type": "hardware",
      "pin_request": {
        "min_length": 6,

```

(continues on next page)

(continued from previous page)

```

        "entry_method": "generated",
        "delivery_method": "email"
    },
    "credentials": [
        {
            "idp_id": "entraid",
            "user_id": "enduser@example.com"
        },
        {
            "idp_id": "pingone",
            "relying_party_id": "pingone.com",
            "user_id": "enduser@example.com"
        }
    ]
}
]
}

```

3.2.1.5 Version 1 - Create Credential Request

Version 1 of the Create Credential Request API is a credential-only request supporting local on-site programming of credentials by enrollment administrators, using a mobile device. The API also supports the option to configure the credential creation flow to allow the end user to enter their PIN through the mobile device used in an on-site enrollment scenario.

Note: Version 1 of the FIDO Connector Create Credential API does not support multiple IdPs. By default the IdP configured in the environment variable `IDP_DEFAULT` will be used. To be able to connect to multiple IdPs, use the *Create Credential Request version 2.0 API*.

Endpoint: POST `/v1/credential-requests`

Request body

```

{
  "recipient": {
    "firstname": "Firstname",
    "lastname": "Lastname",
    "email": "firstname.lastname@example.com"
  },
  "credentials": [
    {
      "user_id": "firstname.lastname@example.com"
    }
  ],
  "pin_request": {
    "min_length": 6
    "entry_method": "user_entered"
  },
  "authenticator_name": "My passkey",

```

(continues on next page)

(continued from previous page)

```

    "customization_id": "ABC123",
    "enrollment_contacts": {
      "emails_to": ["firstname.lastname@example.com"]
    }
  }
}

```

3.2.2 Get Credential Request by ID

VERSION 2.0 (Current)

Provides the ability to view the status of a credential request with a specific ID in the FIDO Connector. The endpoint returns the status of programmed credentials, both programmed by Yubico as well as programmed on-site by enrollment administrators. The credential are returned as an array of **authenticators** and **credentials**.

If the request includes shipment of physical keys, the credential creation status for each YubiKey and the associated IdP included in the shipment are also returned.

Endpoint: GET /v2/credential-requests/{credential-request-id}

Success response body

```

{
  "data": {
    "credential_request_id": "9c8cb30f-bc7a-4e06-bfac...",
    "state": "received",
    "authenticators": [
      {
        "id": "fe0e64f2-eb18...",
        "name": "My YubiKey",
        "state": "processing",
        "credentials": [
          {
            "idp_id": "pingone",
            "relying_party_id": "pingone",
            "user_id": "firstname.lastname@example.com",
            "state": "activation_pending",
            "retryable_until": "2026-09-25T10:00:00Z",
            "error_message": "optional"
          }
        ]
      }
    ]
  },
  "shipment": {
    "shipment_id": "R9SJFBzFbMw...",
    "shipment_items": [
      {
        "id": "1",
        "name": "YubiKey 5 NFC",
        "state": "processing",
        "credentials": [

```

(continues on next page)

(continued from previous page)

```

        "idp_id": "entraid",
        "user_id": "firstname.lastname@example.com",
        "state": "activation_pending",
        "error_message": "optional"
    },
    {
        "idp_id": "pingone",
        "relying_party_id": "pingone",
        "user_id": "firstname.lastname@example.com",
        "state": "activation_pending",
        "error_message": "optional"
    }
]
},
"mailing_address": {
    "street_line1": "#123 Street Name",
    "street_line2": "Unit #456",
    "postal_code": "98765",
    "city": "City Name",
    "region": "MA",
    "country_code": "US"
}
}
}
}

```

Parameters

- **credential_request_id** The credential ID that was returned in the credential request response, see [Create Credential Request](#).
- **state** The state of the credential request can be one of “received”, “started”, “complete”, “partial”, “failed”, or “cancelled”.
- **authenticators** Optional, for example one or more IdPs. If provided cannot be blank.
- **authenticators.id** The ID of the created authenticator.
- **authenticators.name** Required, either a custom name or the default “Pre-Reg YubiKey”.
- **authenticators.state** The state of the credential processing, can be one of “processing”, “complete”, “partial”, “failed”, or “cancelled”.
- **retryable_until** The date when a credential activation will be retried after an activation failure.
- **shipment** Optional, be provided if a physical shipment is part of the request.
- **shipment.shipment_items** Information provided to the IdP when the credential is sent back to the IdP.
- **shipment_items.id** The ID of the shipment of physical items if included, returned in the credential request response, see [Create Credential Request](#).
- **shipment_items.state** The state of the shipment of physical items, if included, can be one of “processing”, “complete”, “partial”, “failed”, or “cancelled”.

Error response body

```
{
  "error_code": "error type",
  "error_message": "error message",
  "error_data": {
    "key": "value"
  }
}
```

Response messages

- 200 Success
 - 404 Not Found
 - 401 Unauthorized on FIDO Connector
 - 500 Unexpected Error or Failure Authenticating with FPR
-

3.2.2.1 Version 1 - Get Credential Request by ID

Provides the ability to view the status of a credential request with a specific ID in the FIDO Connector.

Endpoint: GET /v1/credential-requests/{credential_request_id}

Success response body

```
{
  "credential_request_id": "9c8cb30f-bc7a-4e06-bfac...",
  "credential_request_state": "processing",
  "credentials": [
    {
      "user_id": "firstname.lastname@example.com",
      "relying_party_id": "pingone",
      "idp_id": "pingone",
      "credential_state": "activation_pending"
    }
  ]
}
```

3.2.3 Check Deployment Status**VERSION 2.0 (Current)**

Provides the status of deployment components. The response returns the configuration settings specific to the default IdP configured in the FIDO Connector.

When testing a deployment, you can first do a call to /v2/status to verify that the API is operational and the client can connect to it. It is also a way to ensure that some of the key properties provided during deployment are set.

Endpoint: GET /v2/status

Success Response Body

Example with Entra ID, PingOne PingID and PingOne AIC as IdPs.

```

{
  "AZURE_STORAGE_BLOB_ENDPOINT": "https://...blob.core.windows.net/",
  "AZURE_TABLES_SHIPMENTS_TABLE_NAME": "fprshipments",
  "AZURE_TENANT_ID": "fe0e64f2-eb18...",
  "CRON_PROCESS_SHIPMENT_SCHEDULE": "0 0 * * * *",
  "CRON_PROCESS_CREDENTIAL_REQUESTS_SCHEDULE": "0 */12 * * * 0",
  "CRON_DATA_CLEANUP_SCHEDULE": "0 30 * * * *",
  "CRON_DATA_CLEANUP_COMPLETED_DAYS": "0",
  "FIDO_CONNECTOR_VERSION": "2.0.0-rc5",
  "LOGGING_LEVEL_COM_YUBICO": "DEBUG",
  "EMAIL_API_SEND_ENDPOINT": "https://...westus2.logic.azure.com:443/...",
  "YE_API_BASE_URL": "https://.../v1",
  "YE_JWKS_SIGN_ENDPOINT": "https://.../.well-known/yubico-sign.jwks",
  "YE_JWKS_TRANSPORT_ENDPOINT": "https://.../.well-known/yubico-transport-keys.jwks",
  "RFPR_API_BASE_URL": "https://.../credentials/connector",
  "RFPR_RETRY_NUMBER_DAYS": "1",
  "ENROLLMENT_SUITE_API_BASE_URL": "https://...",
  "IDP_DEFAULT": "ping-aic",
  "IDP_ENABLE_MULTI_IDP": "true",
  "IDP_CONFIGURATION_PATH": "idp-configuration.yaml",
  "IDP_CONFIGURATIONS": {
    "pingone": {
      "default-relying-party": "pingone.com",
      "api-base-url": "https://api.pingone.com/v1",
      "environment-id": "c15efal0-4ebb...",
      "pre-registration-timeout-days": 30,
      "type": "pingone",
      "client-id": "6941c5d6-495b...",
      "auth-base-url": "https://auth.pingone.com"
    },
    "entraid": {
      "api-version": "beta",
      "graph-endpoint": "https://graph.microsoft.com",
      "type": "entraid",
      "challenge-timeout-minutes": 20160,
      "client-id": "b7b32d87-b933...",
      "tenant-id": "fe0e64f2-eb18...",
      "online-endpoint": "https://login.microsoftonline.com"
    },
    "ping-aic": {
      "journey": "WebAuthnPreReg-FIDOConnector",
      "challenge-timeout-milliseconds": 13000000000,
      "type": "ping-aic",
      "api-base-url": "https://example.forgeblocks.com",
      "authentication-header": "oidc_id_token",
      "token-id-2": "IDToken2",
      "token-id-1": "IDToken1",
      "scope": "api://scope-api.onmicrosoft.com",
      "realm": "alpha",
      "client-id": "client-id-string",
      "callback-1": "NameCallback",
      "auth-base-url": "https://example.forgeblocks.com",
      "callback-3": "HiddenValueCallback",

```

(continues on next page)

(continued from previous page)

```
    "callback-2": "MetadataCallback"
  }
}
```

Response messages

- 200 Success
-

3.2.3.1 Version 1 - Check Deployment Status**Endpoint:** GET /v1/status**Success response body**

Example with Microsoft Entra ID as IdP.

```
"AZURE_KEY_VAULT_ENDPOINT":"https://msfpr...azure.net"
"AZURE_TABLES_ENDPOINT":"https://msfpr...windows.net"
"AZURE_TABLES_SHIPMENTS_TABLE_NAME":"fprshipments"
"AZURE_TENANT_ID":"fe0e64f2..."
"CRON_PROCESS_SHIPMENT_SCHEDULE":"0 0 * * * *"
"CRON_DATA_CLEANUP_SCHEDULE":"0 30 * * * *"
"CRON_DATA_CLEANUP_COMPLETED_DAYS":"0"
"EMAIL_API_SEND_ENDPOINT":"https://prod...IhyfXyc"
"ENTRA_FIDO_API_CLIENT_ID": "123abc...",
"ENTRA_FIDO_API_VERSION": "string",
"ENTRA_FIDO_API_CHALLENGE_TIMEOUT_MINUTES": "string",
"LOGGING_LEVEL_COM_YUBICO":"INFO"
"YE_API_BASE_URL":"https://..."
"YE_JWKS_SIGN_ENDPOINT":"https://...sign.jwks"
"YE_JWKS_TRANSPORT_ENDPOINT":"https://...keys.jwks"
"FIDO_CONNECTOR_VERSION":"1.2.0"
```

3.2.4 Process Shipments**VERSION 2.0 (Current)/1.0** (no changes between versions 1 and 2)

Provides the ability to trigger on-demand the process of retrieving shipment responses from the Delivery service and process them. This API is useful if shipment processing needs to be run right away instead of waiting for the scheduled job.

Endpoint v2: GET /v2/operations/process-shipments**Endpoint v1:** GET /v1/operations/process-shipments**Response messages**

- 204 Success, no content body.

3.2.5 Process Credential Request

VERSION 2.0 (Current)

Provides the ability to process and activate the credential request after fulfillment. When activated, the credential is ready to be used for authentication by the end user.

Endpoint: POST /v2/operations/process-credential-requests

Error response body

```
{
  "error_code": "error type",
  "error_message": "error message",
  "error_data": {
    "key": "value"
  }
}
```

Response messages

- 200 Success
 - 401 Unauthorized on FIDO Connector
 - 500 Unexpected Error or Failure Authenticating with FPR
-

3.2.5.1 Version 1 - Process Credential Requests

Endpoint: POST /v1/operations/process-credential-requests

Response messages

- 204 Success, no content body
-

3.2.6 Resend PIN

VERSION 2.0 (Current)

Provides the ability to resend the PIN email for all authenticators and shipment items that have a PIN response available in the Enrollment service.

Endpoint: POST /v2/credential-requests/{credential_request_id}/resend-pin

Error response body

```
{
  "error_code": "error type",
  "error_message": "error message",
  "error_data": {
    "key": "value"
  }
}
```

Response messages

- 200 Success
 - 401 Unauthorized on FIDO Connector
 - 500 Unexpected Error or Failure Authenticating with FPR
-

3.2.6.1 Version 1 - Resend PIN

Endpoint: POST /v1/credential-requests/{credential_request_id}/resend-pin

Provides the ability to resend a PIN if this exists for a credential request. If there are multiple user IDs (userId) in the request, the IDs are provided in the email as a non-URL encoded comma-separated list.

Endpoint: GET /v1/operations/resend-pin/{shipment_id}

Provides the ability to resend the PIN email for a shipment_id. For a shipment request in complete or pending_pin_sending status, this operation retrieves the PIN response from the Delivery service and decrypts it to resend the PIN email.

Response messages

- 204 Success, no content body
-

3.3 FIDO Connector API and FIDO Pre-reg

For *shipment requests* for physical YubiKeys, version 1 of the FIDO Connector extends the `fido2PreRegisteredShipments` in the YubiKey as a Service API, as described in the following. The same POST /v1/fpr/shipments endpoint is called by the FIDO Connector during the workflow processing through the Delivery service.

3.3.1 Create Shipment Request

VERSION 1.0

Endpoint: POST /v1/fpr/shipments

Request body

```
{
  "user_id": "firstname.lastname@example.com",
  "pin_request": {
    "type": "generate",
    "length": 6,
  },
  "enrollment_contacts": {
    "emails_to": ["firstname.lastname@example.com", "itadmin@example.com"]
  },
  "yubico_shipment_request": {
    "reseller_organization_id": "123456",
    "delivery_type": 1,
    "address_validation_bypass": false,
  }
}
```

(continues on next page)

(continued from previous page)

```

"recipient": {
  "recipient_company": "Company name",
  "recipient_email": "firstname.lastname@example.com",
  "recipient_firstname": "First name",
  "recipient_lastname": "Last name",
  "recipient_telephone": "+123456789"
},
"mailing_address": {
  "street_line1": "#123 Street Name",
  "street_line2": "Unit #456",
  "city": "City Name",
  "region": "MA",
  "postal_code": "98765",
  "country_code_2": "US"
},
"shipment_items": [
  {
    "product_id": 1,
    "inventory_product_id": 18,
    "product_quantity": 1,
    "customization_id": "ABC123",
    "fc_additional_data": {
      "yubikey_name": "MyYubiKey"
    }
  }
]
}
}

```

Parameters

- **user_id** Can be provided either as Object ID, for example “user_id”: “123456-abc-123456-xyz”, or as UPN (User Principal Name), for example “user_id”: “firstname.lastname@example.com”.
- **pin_request** Value can be between 4 and 63, inclusive.
- **enrollment_contacts.emails_to** List of strings of email addresses. Provides the ability to enter a list of email addresses to which the PIN can be sent. Using the **enrollment_contacts** object is optional for every IdP implementation.

Note: If using **enrollment_contacts.emails_to**, the default “Send Shipment Pin Azure Logic App” must be appropriately modified to use **emailsTo**. The default reference implementation of “Send Shipment Pin Azure Logic App” sends the PIN to **recipient_email**.

- **reseller_organization_id** Optional, the organization ID of the channel partner (distributor/reseller) in the Delivery service. Use the [Get all channel partners relevant to an org API](#) to retrieve this.
- **recipient_email** The email to receive the PIN (not the principle object name).
- **product_id** Required, ID of the YubiKey product model, see [Product and Inventory Identifiers](#).
- **inventory_product_id** Required, the subscription ID associated with the delivery, see [Product and Inventory Identifiers](#).
- **product_quantity** The number of YubiKeys to include.

- `customization_id` Optional, ID associated with the programming, see [Customizations](#).
- `yubikey_name` Optionally provides the ability to add a custom passkey name in a pre-registration request. The field will be displayed together with the serial number of the YubiKey. For example, if the input name is “My YubiKey” and the serial number is “1234567890”, this displays as “My YubiKey: 1234567890”.

If a custom name is provided, all characters must be ASCII and the field length must not exceed 18 characters.

If a custom name is not provided, the default name “Pre-Reg YubiKey: {SERIAL}” is displayed.

Success response

- 201
- Response body:

```
Created shipment_id from Delivery service
{"data":{"shipment_id":"String"}}
```

Error response

- 401 Unauthorized
- 400 Bad Request
- Response body examples:

```
{"error_code":"ye_error","error_message":"Validation error when creating YED_
↪shipment","error_data":{"code":"validation_error","message":"Input for Last_
↪Name exceeded limit of 20 characters"}}

{"error_code":"api_error","error_message":"PIN `length` must be between 4 and 8
↪","error_data":{"error_type":"validation"}}

{"error_code":"idp_error","error_message":"Could not find user: 7dc95e2f-53-...
↪"}
```

3.3.2 Get Shipment Request Status

VERSION 1.0

Provides the ability to get the processing state of a `shipment_id` created through the Create Shipment Request API.

Endpoint: GET /v1/fpr/shipments/{shipment_id}

Success response

- 200
- Response body:

```
{
  "shipment_id": "R9SJFBzFbMw...",
  "shipment_state": "processing"
}
```

Error response

- Response body:

```
{
  "shipment_id": "string",
  "shipment_state": "string",
  "error_kind": "string",
  "error_message": "string"
}
```

The request can have the following states in the FIDO Connector:

shipment_state	Description
ongoing	The request has been created in the Delivery Service with a Shipment ID. Fulfillment operations and credential creation are in progress with the IdP.
complete	Response from the Delivery Service has been received, the credential has been created on the YubiKey and successfully registered with the IdP.
pending_pin_sending	The credential has been activated but there is a PIN sending failure.

If a processing error has been encountered it will be saved in the **fprshipments** table and returned by the API. Details about the encountered error are provided in **error_kind** and **error_message** as described in the following.

Error	Description
error_kind	This field will contain a string value “GENERAL” if an error has been encountered during processing.
error_message	This field will contain a string value that has the detailed error message returned by the Delivery Service or the IdP.

RELEASE NOTES

This section contains release notes for the FIDO Connector and related APIs, including feature updates, enhancements, resolved issues, and known limitations. The changes reported in the release notes are cumulative.

4.1 2026

4.1.1 Release 2.0.9 (29 September 2026)

Resolved Issues

- **Incorrect payload data validation:** An issue was reported where the request payload contained two credentials with the same `relying_party_id`, `user_id` pair inside the `authenticators` object. An error message was returned, but a credential request had already been created. This issue has been resolved, and proper validation is now performed.

4.1.2 Release 2.0.8 (17 September 2026)

Resolved Issues

- **FIDO Pre-reg order failure due to PIN request validation error:** An issue was reported where the processing of a FIDO Pre-reg shipment request validation failed due to a missing `pin_request.type` parameter. This issue has been resolved, and the `pin_request.type` value now defaults to “generate” if not provided.

4.1.3 Release 2.0.7 (16 September 2026)

New Features & Enhancements

- **Credential processing retry date calculation:** Previously, the date when a credential re-process attempt would occur was calculated only once after the first activation failure. To make the processing more dynamic in cases where the retry configuration has changed, the retry logic has been enhanced. The date of the first failure is now saved, and each re-processing attempt is calculated against the current configuration.
- **Credential processing retry interval:** The default value for re-processing of failed credential activations has been changed from “1 day” to “14 days”. This provides users with additional time to modify settings within the FIDO Connector configuration.
- **“Retry until date” in credential request response:** For enhanced user experience, a new `retryable_until` field has been added to the `GET /v2/credential-requests/:id` endpoint. This allows users to view the last date for when a credential will be retried after an activation failure. See [Get Credential Request by ID](#).

4.1.4 Release 2.0.4 (30 July 2026)

Resolved Issues

- **Error states not used for credential requests:** An issue was reported where v2 credential requests were not using the Azure table “fprshipments” error fields. After creating a shipment request and deleting it, the state in the “fprshipments” table remained in `activation_pending` with no `error` or `error_kind` in the table. This issue has been resolved, and the error fields are now used as expected.

4.1.5 Release 2.0.3 (21 July 2026)

Resolved Issues

- **Credential programming, user.displayName too long:** Some IdPs allow configuration of how the public key credential creation options (PKCCO) `user.displayName` and `user.name` are built, which might result in the exceeding of the maximum number length of 64 characters supported by the YubiKey programming. To avoid this failure, the functionality has been updated so that if the `user.name` and/or `user.displayName` are longer than the limit, a 500 INTERNAL SERVER ERROR will occur displaying a recommendation to change the IdP configuration.

4.1.6 Release 2.0.1 (15 June 2026)

New Features & Enhancements

- **Serial number in Logic App:** A `yubikeySerial` field has been added to the payload sent from the FIDO Connector to the Logic App, allowing for the serial number to be included in the PIN email. The field is separate from the existing `yubikeyDisplayName` field, which remains as-is. As part of this enhancement, `yubikeySerial` has also been added to the Logic App’s request body schema, for usage in the email template.
- **Shipment ID and Credential ID in Logic App:** The `ShipmentId` and `credentialRequestId` fields have been added to the payload sent from the FIDO Connector to the Logic App, including the following:
 - `shipmentId` for v1 FPR Shipments and v2 Credential requests (with shipment).
 - `credentialRequestId` for v1 and v2 Credential requests.

The `resend-pin` endpoint also includes these fields. See [Create Credential Request](#).

- **Validation of single IdP configuration:** Validation has been added to check if the `idp.default` property for a single-IdP instance is mapped to an acceptable value of the IdP implementation, when starting the FIDO Connector.
- **Cleanup of non-completed shipment records:** A new configuration parameter `-CRON_DATA_CLEANUP_PENDING_DAYS`, has been added to the environment variables allowing for stale records to be purged from the Azure Storage Table and Azure Key Vault. See [Environment Variables](#).
- **Multi-IdP configuration file loading:** The ARM Template has been updated with the `multi-idp-configuration-style` parameter for choosing the method to load the multi-IdP configuration YAML file into the FIDO Connector, when enabling multi-IdP support. The parameter accepts the values “none” (default), “volume-mount”, or “blob-storage”. See [Enabling Multi-IdP Support](#).
- **Multi-IdP ARM template parameters:** The new parameters `defaultIdentityProvider` and `ShouldUseMultiIdp` have been added to the ARM template. `defaultIdentityProvider` defines the primary identity provider for the application. `ShouldUseMultiIdp` determines whether the application is configured via environment variables only, or pulls configuration from an external file.

4.1.7 Release 2.0.0 (15 June 2026)

New Features & Enhancements

- **Create credential request (v2):** Implementation of the POST `v2/credential-requests` endpoint for multi-IdP/ multi-credential/ multi-key support. The endpoint creates a request for one or more credentials on one or more IdPs. The request might also include shipments of one or more physical keys, in which case the response includes the `shipment_id`. See [Create Credential Request](#).
- **Get credential request by ID (v2):** Implementation of the GET `v2/credential-requests/{credential_request_id}` endpoint for multi-IdP/ multi-credential/ multi-key support. The endpoint returns the status of programmed credentials, both programmed by Yubico as well as programmed on-site by trusted agents. See [Get Credential Request by ID](#).
- **Credential request activation:** The POST `/v2/operations/process-credential-requests` endpoint has been added to the FIDO Connector API for triggering credential activation. See [Process Credential Request](#).
- **Multi-IdP support:** Multiple IdPs can be configured through a YAML file that can be stored and edited, either using Azure Blob Storage, or by uploading the file to the Azure Blob Storage Container via a volume mount. The ARM template has been updated to include Blob Storage for the multi-IdP YAML configuration file. Validation has been added to check if the `idp.default` property is mapped to one of the provided IdPs. See [Multi-IdP Support](#).
- **Updates to credential request fields for optional/required:** `recipient.telephone` (required), `shipment.mailing_address.region` (optional), `recipient.external_id` (optional), `shipment.customer_reference` (optional), `.shipment_items[*].name` (optional, default “Pre-Reg YubiKey”), `authenticators[*].name` (optional, default “Pre-Reg YubiKey”). See [Create Credential Request](#).

4.1.8 Release 1.2.10 (15 June 2026)

New Features & Enhancements

- **Authentication with YED-API secret:** Previously, the different Yubico services used different authentication secrets. Functionality has been updated so that the YED-API secret is now used to authenticate to all services. See [Authentication](#).
- **Credential request retry configuration:** The configuration for the number of days a credential request will keep retrying after a first failure is now done through the parameter `CRON_PROCESS_CREDENTIAL_REQUESTS_RETRY_NUMBER_DAYS`. See [Environment Variables](#).
- **Credential request resend PIN email:** A new endpoint has been added to v1 of the FIDO Connector. The POST `/v1/credential-requests/{credential_request_id}/resend-pin` endpoint will resend a PIN if this exists for a credential request.

Resolved Issues

- **Resend PIN emails:** The functionality has been enhanced so that the user IDs listed in the PIN email are now sent as a comma-separated list of non-URL encoded `userId`.
- **Credential request PIN email sending:** Previously, when credentials for a request were registered, the credential request state was updated to processed even if sending the PIN email failed. This issue has been resolved. If sending the PIN email fails, then the credential request state remains as processing until the PIN email is successfully sent.

4.1.9 Release 1.2.8 (6 May 2026)

New Features & Enhancements

- **Authenticator name for PingOne AIC:** To make it easier for recipients to identify an authenticator, support has been added for providing a custom authenticator name when using PingOne AIC as IdP. If a custom `authenticators.name` is provided, all characters must be ASCII and the field length must not exceed 18 characters.

If the authenticator is of type “hardware”, the authenticator name together with the YubiKey serial number will be displayed in the IdP. For example, “KeyName: 12345678”. If a custom name is not provided, the default name “Pre-Reg YubiKey: {SERIAL}” is displayed.

4.1.10 Release 1.2.7 (16 April 2026)

New Features & Enhancements

- **Support for remote programming:** The FIDO Connector has been updated to allow remote FIDO Pre-reg programming of YubiKeys by default. Remote programming enables enrollment of YubiKeys by trusted users that securely program credentials on a YubiKey outside of Yubico premises.
- **PingOne AIC credential activation:** In order for the FIDO Connector to support PingOne AIC (Advanced Identity Cloud), a Ping AIC activate credential method has been implemented. This method registers the credentials, and is called by either the activation routine, or the endpoint.
- **Retry logic for activating credentials:** Retry logic has been added in case of failure when registering a YubiKey for an end user account. During the credential activation, if registering the YubiKey fails and the PingOne AIC Journey has returned a new `AuthId`, the retry logic will use that new `AuthId` in the retry operation.
- **Support for end-user provided PIN:** Functionality has been added allowing an end user to provide their own PIN in the remote credential issuance workflow, to avoid sending the PIN via email. In the intended scenario, an IT Admin is processing the credential request with the end user in the same location. If end-user provided PIN is enabled, the enrollment flow will display a prompt, and the device is handed over to the end user who enter their PIN.
- **PingOne AIC default relying party options:** Functionality has been added to make the PingOne AIC default relying party an optional configuration. If `credentials.relying_party_id` is present in the request, then this will be used. Otherwise, if the PingOne AIC default relying party is configured in the FIDO Connector through the `PING_AIC_DEFAULT_RELIVING_PARTY` environment variable, then this will be used. If non of these are provided, the Journey’s Relying-Party Identifier configuration is used.

API updates in this release:

- To support end-user provided PIN, a `PinEntryMethod` parameter to indicate whether the credential request allows a PIN to be entered, has been added to the FIDO Connector API. The value is received by the FIDO Connector, which will let the user enter their PIN in the mobile device if the value is set to “`user_entered`”. If the value is set to “`generated`”, the end-user provided PIN option will not be displayed in the mobile device, and the PIN will be auto-generated.
- The following new APIs are used to create and process *credential requests*:
 - *Create Credential Request*
 - *Get Credential Request by ID*
 - *Process Credential Request*

Note: The current APIs to *request shipment* of enrolled YubiKeys remain the same, if you also want to benefit from the Yubico Delivery service. For more information, see [Create Shipment Request](#).

4.1.11 Release 1.2.3 (23 February 2026)

Resolved Issues

- **Processing of shipments in Delivery Exception state:** Previously, shipments in the Delivery Exception state (106) were treated as an error, causing the job to skip them. This resulted in delays in registering the YubiKey to the user's account, as the next registration attempt was only made once the YubiKey was marked as Delivered. This issue has been resolved. The Delivery Exception state is treated as a valid status, indicating a delivery delay where the shipment is still expected to be delivered.
- **Delegated subnet issue:** An issue was reported where during deployment of the FIDO Connector, an error message "The environment network configuration is invalid: The delegated subnet cannot be used by any other Azure resources." was received. Azure Container Apps require subnet delegation when the "Environment Type" is set to "Workload Profiles". Azure has recently started to set this as the default. To resolve this issue, the deployment template has been modified to include the required subnet delegation.

4.2 2025

4.2.1 Release 1.2.0 (3 November 2025)

New Features & Enhancements

- **Passkey name in FIDO Pre-reg calls:** To make it easier for recipients to identify a passkey, a `yubikey_name` field has been added to the POST `/v1/fpr/shipments` request. The passkey name together with the YubiKey serial number is displayed in the IdP.

The field is optional. If a custom name is not provided, the default name Pre-Reg YubiKey: {SERIAL} is displayed. If a custom name is provided, all characters must be ASCII and the field length must not exceed 18 characters. See [Create Shipment Requests in the API Reference](#).

- **Configurable IdP service:** In order for the FIDO Connector to work with multiple IdPs, support has been added for configuration of IdP-specific application properties. The IdP to be used is set through properties that can be overridden by an environment variable (IDP_DEFAULT). Currently the supported IdPs are Microsoft Entra ID (property value set to `entraid`) and PingOne (property value set to `pingone`). If no value is set for the environment variable (IDP_DEFAULT), the FIDO Connector will use `entraid` as the default.
- **Passkey name provided in PIN email:** To make it easier for recipients to identify passkeys, functionality has been added to provide the passkey name together with the PIN in the email sent to recipients. The payload sent to the Azure Logic App providing the email now includes a new field `yubikeyDisplayName`, which is the authenticator display name. The email also includes the serial number of the YubiKey.
- **PingOne Object_Id (username) as user_id:** As part of the added FIDO Connector PingOne support, functionality has been added to verify that the `user_id` used is a GUID, when creating the MFA User Device on PingOne. If the value is not a GUID, it is handled as an `Object_Id`, and the GUID user ID for it is retrieved from the PingOne endpoint `/environments/{environments}/users`. The `user_id` is then saved as the GUID value.
- **Email sending of PIN to recipient:** The Logic App, which controls the sending of the PIN emails, has been updated to support more IdPs. It is now used only to send emails and does not contain any IdP-specific logic.

By default the email sender Logic App is configured to send PIN email to “recipient_email” provided in the POST /v1/fpr/shipments request.

The POST /v1/fpr/shipments request also has a new `enrollment_contacts.emails_to` field which takes a list of strings with email addresses to which the PIN can be sent. Using the “enrollment_contacts” object is optional for every IdP implementation. When using this method, the default Send Shipment Pin Azure Logic App must be appropriately modified to use `emailsTo`. See Create Shipment Requests in the [API Reference](#).

- **Matching authentication method display names:** Functionality has been added to the FIDO Connector to match the “YubiKey Display Name”, sent in the PIN email, with the Auth Methods “Nickname” available in PingOne.
- **Email functionality:** The default Logic App functionality has been updated to send the PIN email to the email addresses received in the new `emailsTo` list. If no email addresses are provided, sending the PIN email will fail. As part of this, the ARM template has also been updated to support the Logic App updates.

Resolved Issues

- **Resending PIN not possible:** An issue was reported where it was not possible to resend the PIN due to issues with the Logic App. The credentials were activated but the shipment request remained in state “Ongoing”. In this case, the PIN resend functionality expects the state to be “Complete”. This issue has been resolved, and resending a PIN now works as expected.

4.2.2 Release 1.1.6 (16 October 2025)

New Features & Enhancements

- **Expiration date for Azure Key Vault secrets:** A default expiration date has been set for Key Vault secrets in order to align with policies requiring an expiration date. When creating a Key Vault secret, the FIDO Connector will set `SecretProperties.setExpiresOn(OffsetDateTime expiresOn)` to *45 days*. This value can be changed in the Azure portal. For more information, see [Key Vault - Secrets \(Microsoft documentation\)](#).

4.2.3 Release 1.1.4 (22 August 2025)

New Features & Enhancements

- **Get user using UPN (User Principal Name):** Previously, in the JSON request the `user_id` was provided as Object ID. To enhance the user experience, the input now allows `user_id` to be provided either as Object ID, for example `"user_id": "123456-abc-123456-xyz"`, or as UPN, for example `"user_id": "username@yubico123.sample.com"`.
- **Connector container app version:** Functionality has been updated to show the application version. `/v1/status` now returns the `"FIDO_CONNECTOR_VERSION": "string"` field displaying the version of the FIDO Connector application software.

Resolved Issues

- **Connector API response:** Previously the API was returning a “500 Internal Server Error” message, for example when an invalid YubiEnterprise API token was used. To provide more guidance to the user, the error message has been changed to show a “401 Unauthorized” error message instead.

4.2.4 Release 1.1.1 (25 June 2025)

New Features & Enhancements

- **PIN length changed to 4-63.** The validation in the API accepts a PIN length value between 4 to 63, inclusive. This means you can enter any number from 4 up to 63 (including either 4 or 63) as PIN length. See [API Reference](#).
- **Support for address validation override.** An `address_validation_bypass` flag (true/false) has been added to the API. If set to “true”, the API will accept the provided address without further validation. See [Address Validation](#).

4.2.5 Release 1.0.0 (3 April 2025)

First release of the Yubico FIDO Connector App. The application is deployed to a Microsoft Azure subscription and handles most of the Customer Orchestration complexities. For more information about included features, see [Yubico FIDO Connector App](#).

COPYRIGHT

© 2021-2026 Yubico AB. All rights reserved.

5.1 Trademarks

Yubico and YubiKey are registered trademarks of Yubico AB. All other trademarks are the property of their respective owners.

5.2 Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design, and manufacturing. Yubico shall have no liability for any error or damages of any kind resulting from the use of this document.

All Yubico Software referenced in this document is licensed to you under the terms and conditions accompanying the software or as otherwise agreed between you or the company that you are representing.

5.3 Contact Information

Yubico AB
Gävlegatan 22
113 30 Stockholm
Sweden

5.4 Getting Help

Documentation is continuously updated on <https://docs.yubico.com/> (this site). Additional support resources are available in the [Yubico Knowledge Base](#).

Click the links to:

- [Submit a support ticket for YubiKey as a Service and Customer Portal](#)
- [Submit other support requests](#)
- [Contact our sales team](#)

5.5 Feedback

Yubico values and welcomes your feedback. If you think you may have discovered a flaw in our product, please submit a support request at <https://support.yubico.com/s/requests> and provide as much detail as you can.

5.6 Document Updated

2026-09-30 09:34:56 UTC