Netchex API Usage

Third parties can authenticate with external APIs using API Key or OAuth 2.0.

API Key

In each request to the Netchex API, include the following header:

```
Authorization: ApiKey {api-key}
```

Where `{api-key}` is the value of one of the assigned API keys.

OAuth 2.0

Based on Microsoft AD's OAuth 2.0 client credentials flow.

Get an Access Token

Before making a request to the Netchex API, request a token using your client ID and client secret from Netchex's Azure AD tenant:

```
POST /{netchex-ad-tenant}/oauth2/v2.0/token HTTP/1.1          //Line breaks for
clarity
Host: login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded

client_id={client-id}
&scope={netchex-external-api-scope}
&client_secret={client-secret}
&grant_type=client_credentials
```

- `{netchex-ad-tenant}`: the Netchex AD tenant
  - Production value: `9a2eb4be-01ad-4337-8e3d-b92c8166ef2a`
- `{netchex-external-api-scope}`: the default scope for the Netchex external API
  - Production value: `https://primaryauth.onmicrosoft.com/be3d5ddc-fed7-4042-a4b6-29155e2ea60e/.default`
  - Append `/.default` to the application ID URI of the Netchex external API registration
- `{client-id}`: the value of your client ID
- `{client-secret}`: the value of your client secret

A successful response looks like this:

```
{
  "token_type": "Bearer",
```

```
  "expires_in": 3599,
  "access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Ik1uQ19WWmNBVGZNNXBP..."
}
```

Use the value of the `access_token` property in subsequent requests.

Use an Access Token

In each request to the Netchex API, include the following header:

```
Authorization: Bearer {access-token}
```

Where `{access-token}` is the value of the access token.

Events

Netchex uses Azure Event Grid to send webhook events. Events are POSTed as JSON HTTP requests to your selected endpoint with the following format:

```
[
  {
    "topic": string,
    "subject": string,
    "id": string,
    "eventType": string,
    "eventTime": string,
    "data":{
      object-unique-to-each-event-type
    },
    "dataVersion": string,
    "metadataVersion": string
  }
]
```

Property	Type	Description
`topic`	`string`	unique identifier for this event stream
`subject`	`string`	unique identifier for the subject of the event
`id`	`string` in .NET Guid format `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`	unique identifier for this event
`eventType`	`string`	unique identifier for this event type

Property	Type	Description
`eventTime`	`string` in ISO 8601 format	date and time of the event
`data`	`object`	custom metadata object for the event - see Event Types
`dataVersion`	`string`	schema version of the event message format
`metadataVersion`	`string`	schema version of the event metadata format

Event Message Authentication

You can authenticate the sender of events using a shared secret embedded in a query parameter.

The query parameter will be `token`, and the value of the parameter will be unique to your account. Check the value on each POST to authenticate.

Endpoint Validation

Before receiving events, you must validate your webhook endpoint. At the time that the webhook endpoint is registered, you'll receive a validation event with the following format:

```
[
  {
    "id": "",
    "topic": "",
    "subject": "",
    "data": {
      "validationCode": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
      "validationUrl": "https://validation-url.com/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"
    },
    "eventType": "Microsoft.EventGrid.SubscriptionValidationEvent",
    "eventTime": "",
    "metadataVersion": "1",
    "dataVersion": "1"
  }
]
```

To prove ownership of the endpoint, echo back the validation code in this format:

```
{
  "validationResponse": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```

You must respond with the **HTTP 200** response code.

Event Types

Each event type is listed below with its properties in list form below it.

Event type: `companyPayrollInvoiced`

- `payrollId`: `int32` as `number`
- `companyId`: `int32` as `number`

Event type: `employeeAdded`

- `employeeId`: `int64` as `number`
- `companyId`: `int32` as `number`

Event type: `employeeBenefitCoverageEnded`

- `employeeBenefitId`: `int64` as `number`
- `employeeId`: `int64` as `number`
- `companyId`: `int32` as `number`

Event type: `employeeBenefitEnrolled`

- `employeeBenefitId`: `int64` as `number`
- `employeeId`: `int64` as `number`
- `companyId`: `int32` as `number`

Event type: `employeeBenefitUpdated`

- `employeeBenefitId`: `int64` as `number`
- `employeeId`: `int64` as `number`
- `companyId`: `int32` as `number`

Event type: `employeeDependentAdded`

- `dependentId`: `int64` as `number`
- `employeeId`: `int64` as `number`
- `companyId`: `int32` as `number`

Event type: `employeeDependentUpdated`

- `dependentId`: `int64` as `number`
- `employeeId`: `int64` as `number`
- `companyId`: `int32` as `number`

Event type: `employeeEnrollmentCompleted`

- `employeeId`: `int64` as `number`
- `companyId`: `int32` as `number`

Event type: `employeeTerminated`

- `employeeId`: `int64` as `number`
- `companyId`: `int32` as `number`

Event type: `employeeUpdated`

- `employeeId`: `int64` as `number`
- `companyId`: `int32` as `number`