Build a custom connector with API key authentication to retrieve exchange rates

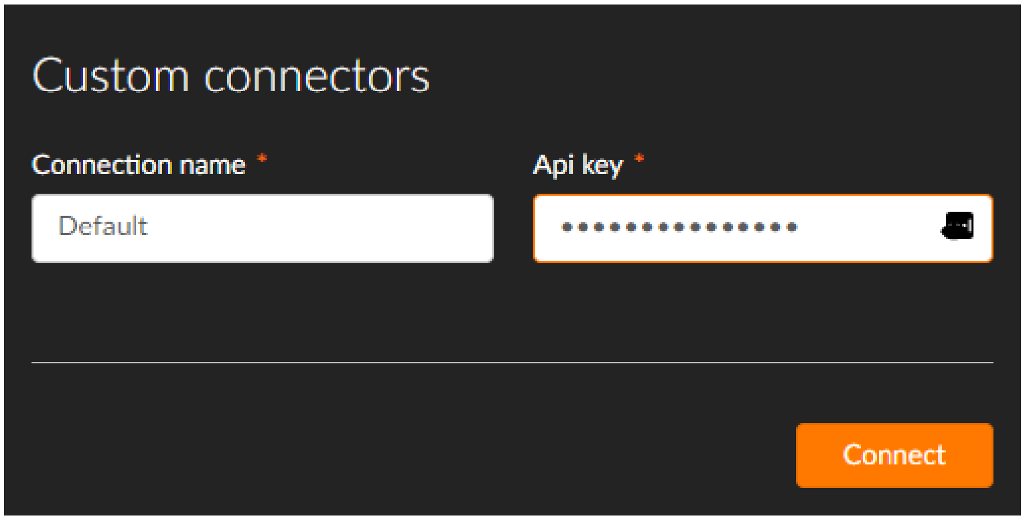
In this example, you will create a custom connector that uses API key authentication to use current exchange rates to approve or reject purchase order requests.

The complete OpenAPI Specification and icon for this example are available here.

Summary

API key authentication requires Nintex Workflow Cloud to provide a secret security token when making the request of the API, which must be accepted by the API for the API to process the request. The token can be provided either in the request header, or in the query.

When using custom connectors with API key authentication, you must create a connection that stores the security token so it can be passed to the API with the requests.



Custom connectors

Connection name *: Default

Api key *: ••••••••••••••••

API key authentication in the OpenAPI Specification

To add API key authentication to an OpenAPI Specification, you:

- Sign up for an API key with the service you want to use
- Define the API key authentication object inside the **securityDefinitions** object
- Reference this API key authentication object inside the HTTP method objects that require authentication

Acquire a security token for the API

The API uses a security token to identify who is making the operation call. You will provide Nintex Workflow Cloud with this token when you create a connection to this custom connector. To acquire your security token for this example:

1. Sign up to OpenExchangeRates.org for the Forever Free plan at this link.
2. Follow the instructions on the Open Exchange Rates page to **Get your API key**.
3. Record the **Name** and the **App ID** somewhere safe: it identifies you to the OpenExchangeRate.org API, and should be kept secret.

Define API key authentication

Step 1: Create the basic OpenAPI Specification

Create an OpenAPI Specification that:

- uses the host **openexchangerates.org**.
- accepts **HTTPS** and produces **application/json**.
- has a **get** operation to **/latest.json** that passes the base currency to provide exchange rates for in the query.
- responds **200** with a schema that returns the base selected currency and the rate to the currencies in USD, GBP, EUR and AUS.
  In this instance, the API actually provides more currencies than these four. To keep the Workflow designer manageable, we are limiting our definition to just the currencies we want to use.

Note: To ensure users only submit valid currency codes to the API, use an enum to pass the base currency parameter.

Restricting parameter options with enums

Copy

```
{
  "swagger": "2.0",
  "info": {
    "version": "1.0.0",
    "title": "Currency Rates"
  },
  "host": "openexchangerates.org",
  "basePath": "/api",
  "schemes": [
    "https"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/latest.json": {
      "get": {
        "summary": "Get exchange rates",
        "description": "Get exchange rates",
        "operationId": "testBasicAuth",
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "base",
            "in": "query",
            "required": false,
            "type": "string",
            "enum": [
              "USD",
              "GBP",
              "EUR",
              "AUD"
            ]
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/rateResult"
            }
          }
        }
      }
```

```
        }
      },
      "definitions": {
        "rateResult": {
          "type": "object",
          "properties": {
            "base": {
              "type": "string"
            },
            "rates": {
              "type": "object",
              "properties": {
                "USD": {
                  "type": "number"
                },
                "GBP": {
                  "type": "number"
                },
                "EUR": {
                  "type": "number"
                },
                "AUD": {
                  "type": "number"
                }
              }
            }
          }
        }
      },
    }
```

Step 2: Create the security definitions object.

Create the **securityDefinitions** object at the end of the OpenAPI Specification, after the **definitions** object but before the final closing brace. Make sure you add a comma to the end of the **definitions** object.

Copy

```
       "definitions": {
        "rateResult": {
          "type": "object",
          "properties": {
            "base": {
              "type": "string"
            },
            "rates": {
              "type": "object",
              "properties": {
                "USD": {
                  "type": "number"
                },
                "GBP": {
                  "type": "number"
                },
                "EUR": {
                  "type": "number"
                },
                "AUD": {
                  "type": "number"
                }
              }
            }
          }
        }
      },
      "securityDefinitions": {

      }
    }
```

Step 3: Create the API key authentication definition

Inside the **securityDefinitions** object, create an object to define the API key authentication. You can name the security object any name that is unique within the OpenAPI Specification. In this example, we have named it **myApiKey**.

Add a **type** with a value of **apiKey** to the object to define it as API key authentication.

Copy

```
    "securityDefinitions": {
      "myApiKey": {
        "type": "apiKey",

      }
    }
```

Step 4: Add the API authentication parameter details

Inside the **myApiKey** object, add a **name** key for the parameter to hold the security token. The value must be the **parameter** name defined by the API.

Add the location it will be passed in with the **in** key.

Copy

```
    "securityDefinitions": {
      "myApiKey": {
        "type": "apiKey",
        "name": "app_id",
        "in": "query"
      }
    }
```

Step 5: Add a security array to the HTTP method object

Inside the HTTP method object, create a **security** array.

Copy

```
    "paths": {
      "/latest.json": {
        "get": {
          "summary": "Get exchange rates",
          "description": "Get exchange rates",
          "operationId": "testBasicAuth",
          "produces": [
            "application/json"
          ],
          "security": [

          ],
```

Step 6: Add the security object reference to the array

Inside the array, reference the security object you defined earlier by using its name as the key, and opening and closing brackets as the value.

Copy

```
    "paths": {
      "/latest.json": {
        "get": {
          "summary": "Get exchange rates",
          "description": "Get exchange rates",
          "operationId": "testBasicAuth",
          "produces": [
            "application/json"
```

```
],
"security": [
  {
    "myApiKey": []
  }
],
```

The OpenAPI Specification

This is the complete OpenAPI Specification that uses API key authentication to retrieve exchange rates.

Copy

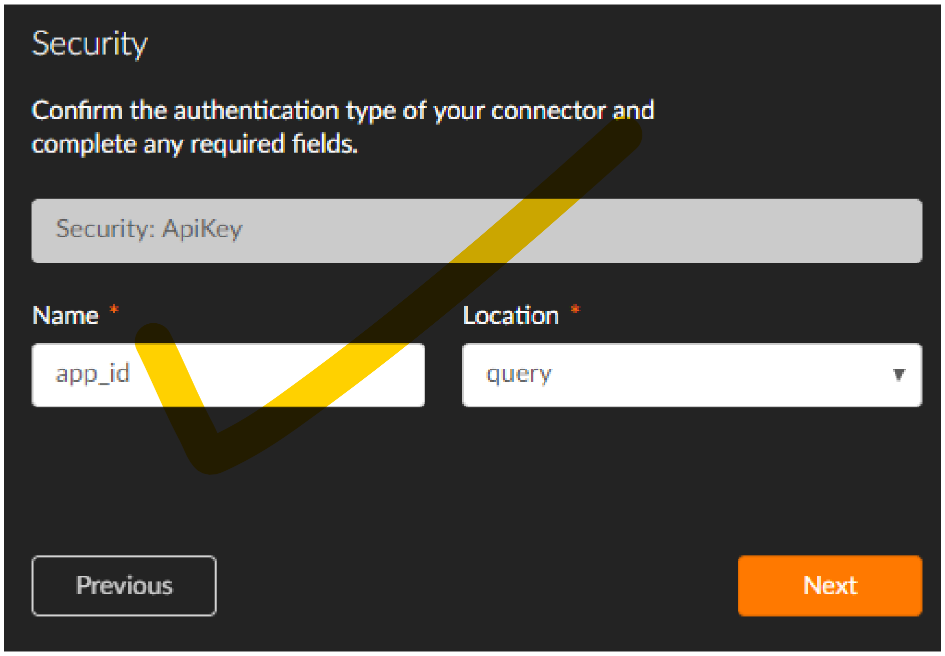
```
{
  "swagger": "2.0",
  "info": {
    "version": "1.0.0",
    "title": "Currency Rates"
  },
  "host": "openexchangerates.org",
  "basePath": "/api",
  "schemes": [
    "https"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/latest.json": {
      "get": {
        "summary": "Get exchange rates",
        "description": "Get exchange rates",
        "operationId": "testBasicAuth",
        "produces": [
          "application/json"
        ],
        "security": [
          {
            "myApiKey": []
          }
        ],
        "parameters": [
          {
            "name": "base",
            "in": "query",
            "required": false,
            "type": "string",
            "enum": [
              "USD",
              "GBP",
              "EUR",
              "AUD"
            ]
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/rateResult"
            }
          }
        }
      }
    }
  },
  "definitions": {
    "rateResult": {
      "type": "object",
      "properties": {
        "base": {
          "type": "string"
        },
        "rates": {
          "type": "object",
          "properties": {
            "USD": {
              "type": "number"
            },
            "GBP": {
              "type": "number"
            },
            "EUR": {
              "type": "number"
            },
            "AUD": {
              "type": "number"
            }
          }
        }
      }
    }
  },
  "securityDefinitions": {
    "myApiKey": {
      "type": "apiKey",
      "name": "app_id",
      "in": "query"
    }
  }
}
```

Create the purchase approval workflow

Step 1: Add your custom connector

Import the OpenAPI Specification you created into Nintex Workflow Cloud:

1. Open your Nintex Workflow Cloud tenancy.
2. Click **Xtensions** in the dashboard to open the Xtensions page.
3. Click ➕ in the Custom connector list.
4. Click **Choose a file**.
5. Navigate to the OpenAPI Specification on your computer.
6. Wait for Nintex Workflow Cloud to validate the file.
7. Click **Next**.
8. Nintex Workflow Cloud detects the API key security template.



Security

Confirm the authentication type of your connector and complete any required fields.

Security: ApiKey

Name * app_id

Location * query

9. Click **Next**.
10. Edit the **Name** of the connector, which becomes the name of the action group in the Workflow designer.
11. Edit the **Description** of the connector. This appears in the Custom connector list in the Xtensions page.
12. Select or upload an icon for the connector. This is displayed for each action or event in the Workflow designer
13. Click **Publish**.

Step 2: Create the workflow

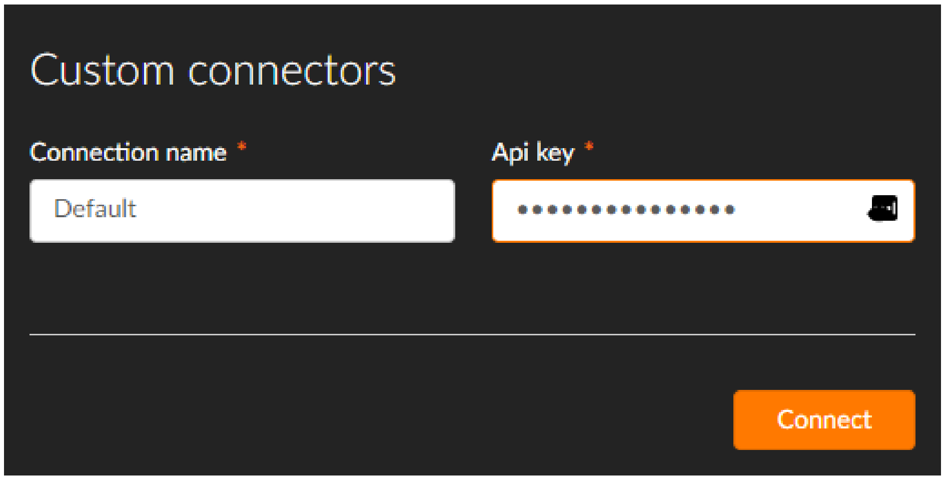
The workflow will retrieve the currency conversion rates based on the United States dollar and compare the AUD rate against a threshold level to recommend whether purchases in Australian dollars be approved.

To create the workflow:

1. Click **Create workflow** in your Nintex Workflow Cloud tenancy.
2. Configure the **Start event** to be a **Public web form** with two variables:
   - Email (text)
   - Purchase request (text)
   - Purchase amount in AUD (decimal)

For more information on designing forms in Nintex Workflow Cloud, see Design a form.

3. Drag the **Get exchange rates** action from the action toolbox to after the **Start event**.
4. Click the **Connection** field and select **Add new connection**.
5. Type the **Name** you recorded from OpenExchangeRates.org in the **Connection name** field.
6. Type the **App ID** you recorded from OpenExchangeRates.org in the **Api key** field.

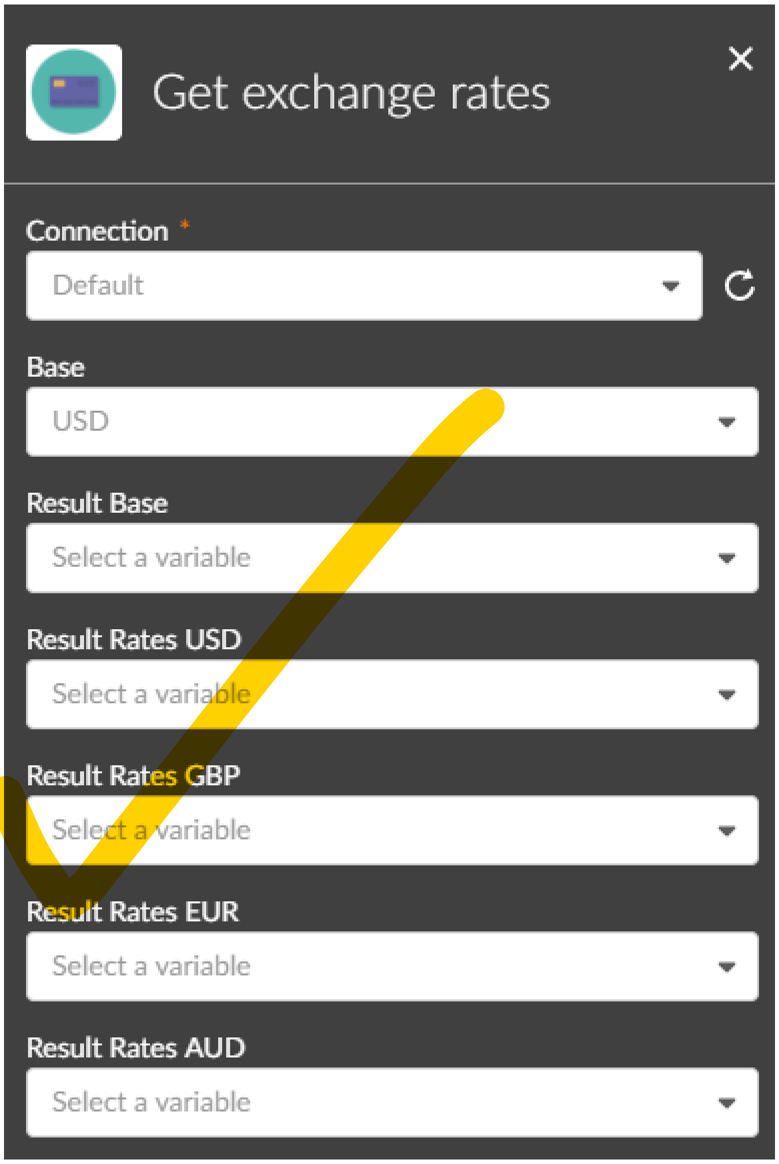


Custom connectors

Connection name *: Default

Api key *: •••••••••••••••

7. Select **USD** in the **Base** configuration field.



Get exchange rates

Connection *: Default

Base: USD

Result Base: Select a variable

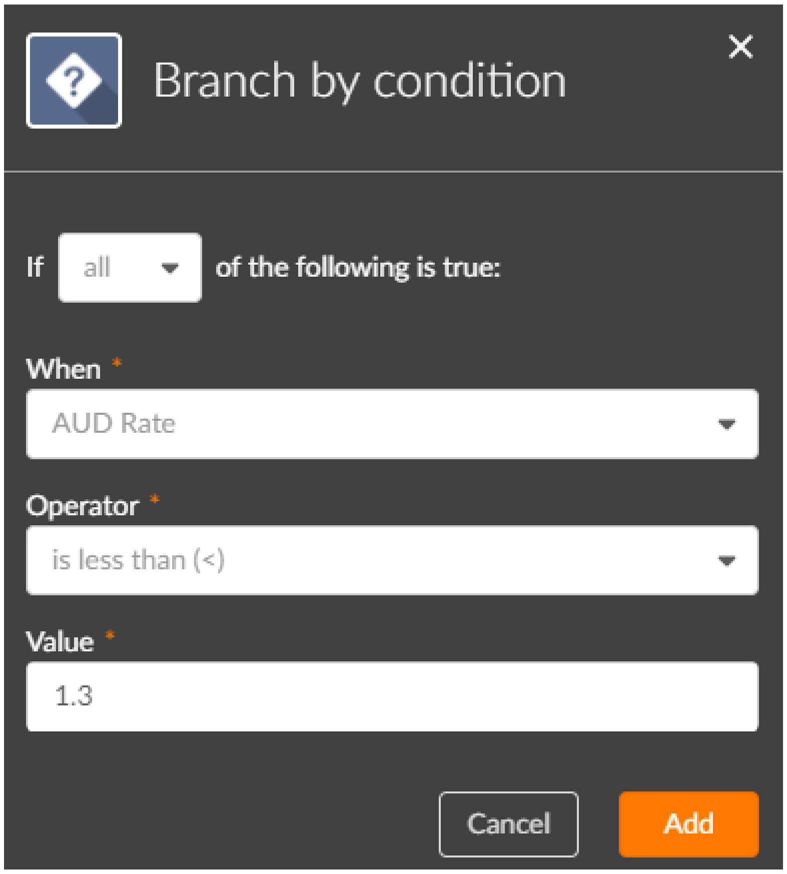
Result Rates USD: Select a variable

Result Rates GBP: Select a variable

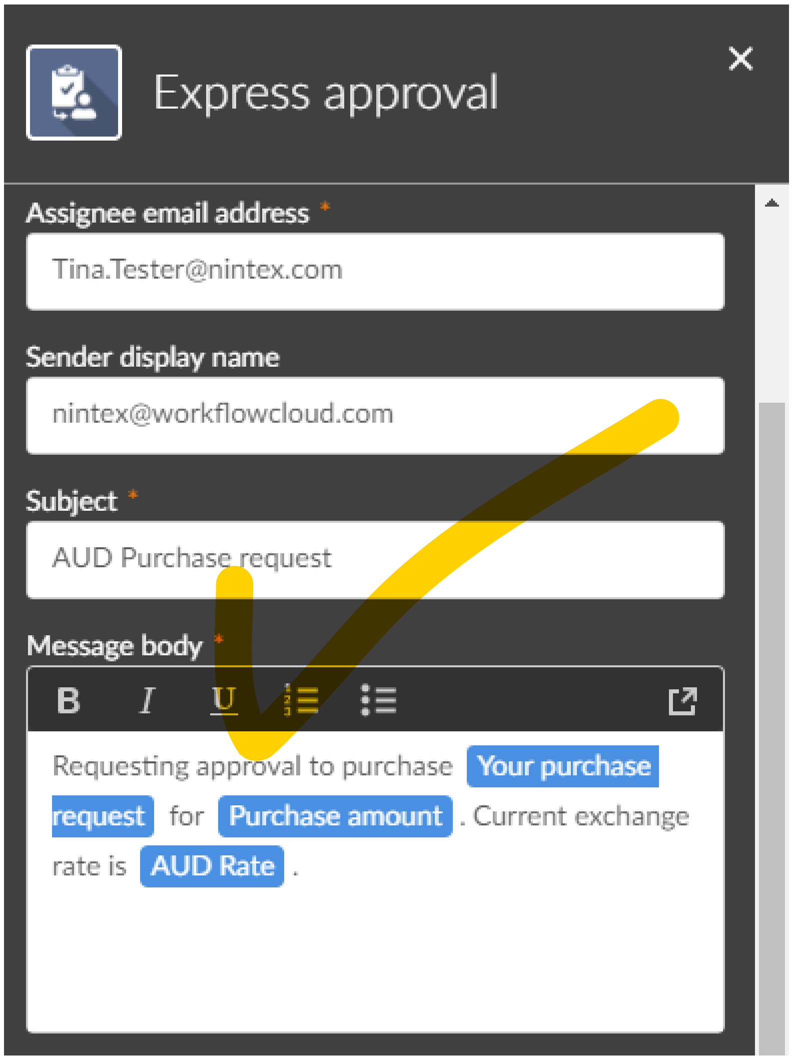
Result Rates EUR: Select a variable

Result Rates AUD: Select a variable

8. Create a decimal variable to store the **Results Rates AUD** return value.
9. Drag a **Branch by condition** action after the **Get exchange rates** action.
10. Configure the **Branch by condition** action to test whether the **AUD rate** is less than **1.3**.



11. Drag a **Set a variable value** action to the **No** branch, and configure it to set a message variable that says the purchase has been pre-approved.
12. Drag an **Express approval** action below the **Set a variable value** in the No branch, and configure it to send the purchase request information and the current exchange rate to the financial officer for approval.



Express approval

Assignee email address *

Tina.Tester@nintex.com

Sender display name

nintex@workflowcloud.com

Subject *

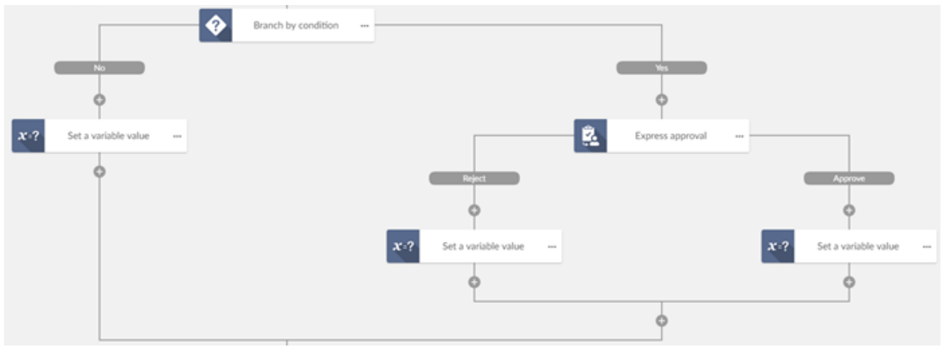
AUD Purchase request

Message body *

Requesting approval to purchase Your purchase request for Purchase amount . Current exchange rate is AUD Rate .

Note: For testing purposes, use your own email address.

13. Drag a **Set a variable value** action to the **Reject** branch, and configure it set a message variable to say the purchase request has been rejected by the financial officer.
14. Drag a **Set a variable value** action to the **Approve** branch, and configure it set a message variable to say the purchase request has been approved by the financial officer.

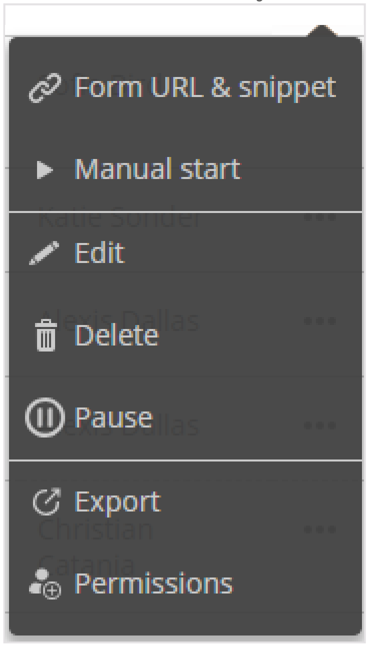


15. Drag a **Send an email** action after the branch converges.
16. Configure the **Send an email** action to email the message variable.
17. Publish the workflow.

Tip: If you want to troubleshoot a custom connector, select **Development** as the Assigned Use when you publish the workflow. Development workflows display more detailed error messages in their instance details. Republish your workflow as Production when you're ready to use it.

Step 3: Test the workflow

1. Click **Workflow** in your Nintex Workflow Cloud tenancy.
2. Click ··· on the workflow you created.



3. Click **Manual start**.
4. Type your email address, purchase order request and the purchase amount in the form.
5. Click **Start** to receive either:
    - An email approving your purchase.
    - An email requesting approval from the financial officer with the purchase details.