PingOne DaVinci

July 2, 2025

PINGONE DAVINCI

Table of Contents

PingOne DaVinci

PingOne DaVinci is an orchestration platform that lets you create flows using connections and logical operators. These flows guide users through defined processes that can present customized pages, modify values, or perform other actions.

Release Notes

- Release Notes

Get Started With DaVinci

- Introduction to PingOne DaVinci
- Getting started with DaVinci
- How to create a flow
- DaVinci Best Practices

Use DaVinci

- Use Cases
- Connectors
- Launching a PingOne flow with a redirect using an external IdP
- Integrating Flows into Applications

Learn More

- DaVinci Community
- DaVinci Support
- DaVinci customer training (existing customers only)
- Partner Portal (partners)

Release Notes

New features and improvements in PingOne DaVinci. Updated June 23, 2025.

June 2025

June 23

Custom Analytics improvements

Improved

We've added new options to the Custom Analytics display in the dashboard. You can now create graphs comparing flows or node outcomes over time. Learn more in Dashboard.

June 13

Flow validation improvements

Improved

We've improved the flow validation feature to enable automatic validation whenever you save a flow and to clearly associate validation results with specific flow versions. Learn more in Validating a flow.

June 11

Flow canvas improvements

Improved

We've made improvements to the flow canvas. When you open a flow, there's now a search option that lets you locate specific nodes using connector or node information. Learn more in Using flow search.

June 9

The PingOne connector has been updated

Info

Learn more in the PingOne connector release notes.

May 2025

May 27

Improved the PingOne Forms feature

Improved

We've made several improvements to the PingOne Forms feature, focusing on multi-factor authentication (MFA), agreements, and improving the experience for administrators and end users:

- Added new form templates
- Added new components, including checkboxes, agreements, and polling
- Enabled dynamic text within fields

Learn more in the PingOne forms documentation.

May 23

Added flow validation tool

New

We've added a tool for validating flows. The tool identifies errors that will prevent a flow from completing successfully and provides warnings for other issues. Learn more in Validating a flow.

May 20

Flow HTTP Response Timeout accepts invalid values

Issue DV-18324

You can currently enter **Flow HTTP Response Timeout (in seconds)** values above the maximum value of 120 seconds. If you specify a value above the maximum, the minimum value of 15 seconds is used.

The workaround is to specify a value between 15 and 120 seconds.

May 19

Added subflow error handling for flow export

New

We've added error checks to the flow export process. When you use the **Download Flow JSON** option and select the **Include Subflows** option, an error message displays if your **Flow Conductor** connectors use invalid subflows or subflow versions. Learn more in Exporting a flow.

April 2025

April 29

Improved handling of user language preferences

Improved

DaVinci now serves localized content by matching the end user's language preferences more intuitively and consistently.

The LexisNexis connector has been updated

Info

Learn more in the LexisNexis connector release notes.

External authentication through Microsoft Entra ID is now available

New

You can now use Microsoft Entra ID users to leverage external authentication providers for multi-factor authentication (MFA). You can enable this integration using DaVinci flows and flow policies.

The Setting up PingOne SSO, DaVinci, and PingID as the external MFA provider for Microsoft Entra ID use case explains the configuration and includes two sample flows to simplify the integration process.

In PingOne Forms, the legacy Rich Text component is being deprecated

Info

The **Rich Text** component is being deprecated soon and will continue to work for a limited time only. Update your forms to use the Translatable Rich Text component as soon as possible. The new component benefits from upcoming features such as language keys, dynamic text, and mobile SDK support.

April 11

Improved handling of sensitive fields in analytics

Improved

We've improved the handling of sensitive fields in DaVinci analytics by automatically filtering out fields known to be sensitive as well as fields containing keywords, such as `password`, `key`, `secret`, `token`, and `authorization`.

April 10

Added ability to check session in API flows

Improved

We've added the ability for flows launched from the API to process a `global.sessionToken` that is passed in as part of the request body.

March 2025

March 13

The DaVinci configuration panel is now wider

Improved

We've increased the width of the right side panel in the DaVinci canvas. This makes working on node configurations and flow settings more comfortable, especially in code fields and tables with key-value pairs.

March 12

Widget flows can include the PingOne session cookie

Improved

We've improved the widget invocation method so that widget flows can recognize the PingOne session cookie if the `includeHttpCredentials` property is set to `true`.

March 3

Improved flow error message

Improved

We've improved the error message displayed to end users if a flow can't complete successfully, making it more visually appealing and easier for end users to understand.

You can also customize the logo displayed on this error message. Learn more in Editing flow settings.

February 2025

February 26

The PingOne Authentication connector has been updated

Info

We've updated the PingOne Authentication connector. When authenticating users by redirecting the browser to your DaVinci flow, the DaVinci flow policy returns additional attributes to PingOne. You can now override the default format of those attributes.

Learn more in the PingOne Authentication connector release notes.

February 24

Improved dashboard

Improved

We've made multiple improvements to the DaVinci dashboard:

- A new chart displays information about flow executions using the SDK.
- Two new charts show the distribution and trends of flow execution methods.
- A new pie chart in **Custom Analytics** compares flow outcomes for the selected flow.

Learn more in the Dashboard documentation.

February 18

Improved logging of ID values

Improved

We've updated logging so that DaVinci logs and webhook events now consistently include `TransactionId`, `correlationId`, `externalTransactionId`, `sessionId`, and `externalSessionId` at any log level. These attributes are also captured in scenarios such as Login with External IdP and Device Authorization flows to ensure comprehensive traceability.

February 10

Improved language support in DaVinci connectors

New

You can now use the PingOne language feature that enables the use of translatable keys in customizable HTML templates and error connectors. This lets you display a translated version of user-facing content based on the user's browser language setting.

Learn more in Setting up DaVinci Forms for multi-language support.

January 2025

January 13

Add labels to dividers in PingOne Forms

Improved

In PingOne Forms, you can now add optional text to divider to provide context to your users. Align the label left, center, or right to suit your theme.

January 10

Flow policies can check for existing sessions

Improved

We've added an ability for PingOne flow policies to check for existing sessions before launching a flow and to skip the flow if a current session exists. Learn more in Configuring a flow policy.

January 8

Forms now lets you override individual styles from your theme

Improved

In PingOne Forms, you can override your default theme colors for a particular button or link. Previously, this completely reset the appearance and you had to provide all new values. Now, enabling Override Default Styles doesn't affect the look until you change a value, and you can override some colors while keeping others linked to the theme.

December 2024

December 13

Improved PingOne Forms user experience

Improved

We've made a set of user experience improvements for PingOne Forms to improve readability and usability for all users and follow Web Content Accessibility Guidelines. These improvements include alternate text, coloration, and organizational improvements.

December 4

Improved audit logging for PingOne Authentication connector changes

Improved

We've added additional logging for the PingOne Authentication connector, which occurs when a DaVinci flow administrator creates or modifies a PingOne Authentication connector node that uses one of the following capabilities:

- Return Success Response (Redirect Flows)
- Return Success Response (Widget Flows)
- Create or Update Session

If the change results in new values for the User ID, Authentication Methods, or Custom Authentication Methods fields, an audit log event is generated, showing the old and new field values.

November 2024

November 11

Added new Custom Analytics

We've improved the **Custom Analytics** section of the dashboard. Now you can compare data for **Flow Analytics** connector outcomes logged within a flow.

Learn more in the Dashboard documentation.

November 4

Improved CORS handling

Improved DV-2256

We've improved the CORS handling in DaVinci. When you add a flow's launching domain to an application in PingOne, DaVinci uses that domain in its CORS settings. Learn more in Integrating Flows into Applications.

October 2024

October 23

Fixed flow instance variable value preservation

Fixed DV-14812

We've fixed an issue that caused the preset value of flow instance variables to be ignored whenever any other flow instance variable's value was updated.

October 18

Expanded Data Sent to Webhooks by the Flow Analytics Connector

Fixed DV-10798

The Flow Analytics Connector now sends the contents of the **Outcome Status Detail** and **Outcome Description/Comment** fields to webhooks, providing external logging systems with more thorough information about flow errors. You can also now populate these fields with variables or data from previous nodes using the **Variables ({ })** icon.

October 8

Added ability to create a flow from a template

New DV-5239

We've added the ability to create a new flow based on a template. When you click **Add Flow**, you can now create the new flow using a template from the Ping library, using an imported flow JSON, or using a blank page.

Learn more in Using DaVinci flow templates.

September 2024

September 26

The Token Management connector has been updated

Info

Learn more in the Token Management connector release notes.

September 16

Fixed audit trail display for variables

Fixed DV-2022

We've fixed a display issue that caused variables to be labeled as "constructs" in the audit trail.

August 2024

August 27

Added search options for flow analytics

Improved DV-5272

The **Flow Analytics** feature now includes options to search for specific flows by user ID, user name, and correlation ID. The new search parameters can find information from the following connectors:

- Token Management
- PingOne SSO
- PingOne Authentication
- PingOne MFA
- User Policy
- OIDC & OAuth IdP
- PingOne Verify

Learn more in Viewing flow analytics.

July 2024

July 31

Use new "Secret" variables for sensitive data in your Make REST API Call nodes

New DV-5389

Variables are a great way to centrally manage values used in your nodes, including in the HTTP connector's Make REST API Call capability. To help protect sensitive values such as API keys, client secrets, and tokens you use in these nodes, we've added a new **Secret** variable type. For more information, see Variables.

July 23

Audit logs now note addition of custom JavaScript

Improved DV-12469

When custom JavaScript is added to a flow or connector, an audit event is now logged. This enhances security and compliance by providing a clear view of all custom script use, ensuring that modifications can be tracked and audited. See Audit Trail for more information.

July 22

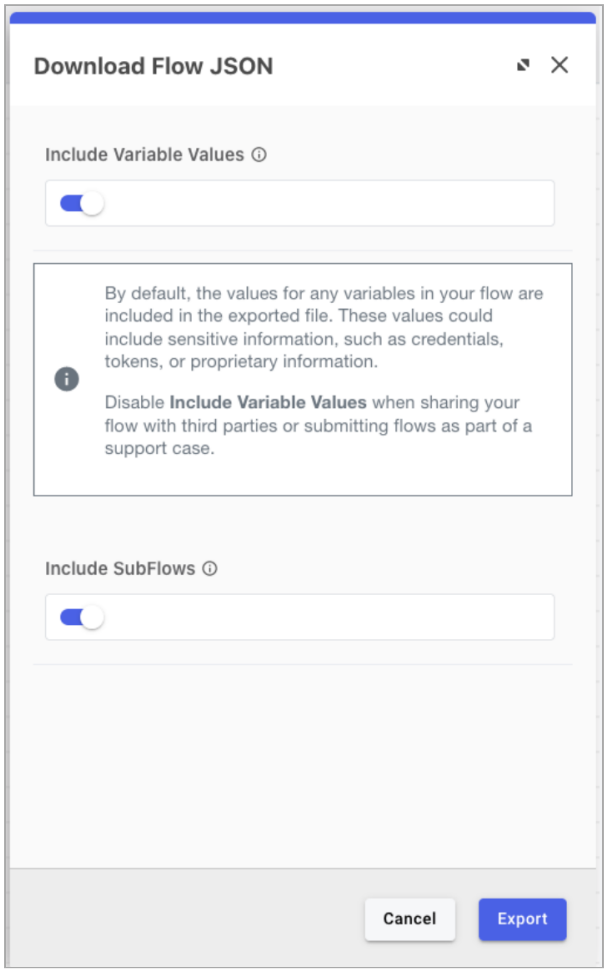
Added global debug logging control

New DV-9426

You can now enable or disable debug level logging for an entire environment. See Company Settings for more information.

Export your flows without variable values

New DV-12440

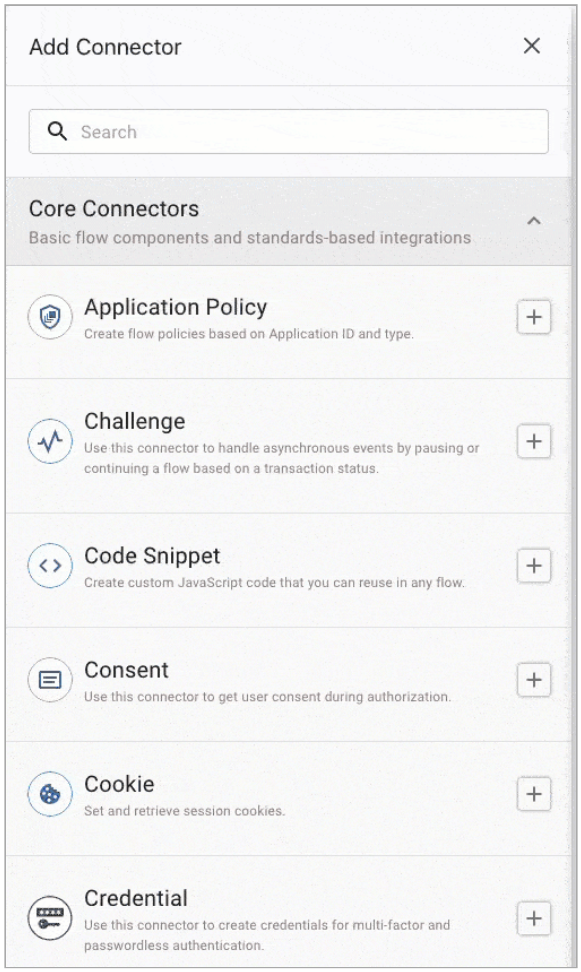


It's important to keep your data safe, so we've added a new **Download Flow JSON** view that gives you the option to exclude variable values when you export a flow. This lets you share flows and get support knowing that proprietary and sensitive variable values never leave your environment. For more information, see Exporting a flow.

July 15

Discover connectors with the refreshed "Add Connector" view

New



When adding a new connector from the **Connectors** view, connectors are now sorted alphabetically and presented in three groups:

- Core Connectors (Basic flow components and standards-based integrations)
- Ping Identity Connectors (Ping Identity products)
- Service Connectors (Third-party platforms and services)

These categories make it easier to find the right type of connector more quickly and eliminates the guesswork required with the previous category system.

This view has also received a visual refresh to give you a more consistent experience with other parts of DaVinci and PingOne.

If there are multiple connectors from the same vendor, they appear in a group under the vendor name. For example, the Amazon DynamoDB and AWS Lambda connectors appear together in the "Amazon" group.

PingOne DaVinci Early Access Features

This section provides early access documentation for the DaVinci features available to customers who opt in to preview new functionality through PingOne.

Early access features can be enabled only at the environment level. You can't enable an early access feature for an entire organization.

Not all features are enabled for early access. Additionally, early access features related to services that aren't in the environment or that aren't allowable by the license assigned to the environment aren't available for opt-in.

Learn more in Managing opt-ins for early access features in PingOne.

Important

Early access features are provided for preview purposes only and are not covered under standard Support SLAs. You can open support cases for feedback, bug reports, configuration questions, or other inquiries related to early access features, but resolution times for these cases will vary. These cases often require collaboration with our Engineering and Product teams, so timelines might exceed the usual SLAs for your Support package.
Topics for these features are draft documentation for early access purposes only and are not complete or final.

Usage Terms

By using the PingOne DaVinci platform, you agree to the following usage terms.

- We reserve the right to pause a flow execution that is deemed unsafe to the customer and the platform.
- If a third-party connector or flow becomes unsafe due to a third-party change, we reserve the right to disable them and provide appropriate communication.
- Rate limiting on APIs is currently in place, and we reserve the right to change those limits over time.
- Any usage that exceeds the purchased subscription allotment is subject to an overage fee.

DaVinci browser support

PingOne DaVinci is supported on the following platforms.

> **Note**
>
> DaVinci does not support Microsoft Internet Explorer 11, including Microsoft applications that use WebView.

> **Note**
>
> Scripts or custom CSS included in flows may limit support.

Browser	Version
Google Chrome	Three most recent major versions
Mozilla Firefox	Three most recent major versions
Apple Safari for MacOS	Two most recent major versions
Microsoft Edge	Two most recent major versions

Introduction to PingOne DaVinci

PingOne DaVinci is an orchestration platform that lets you create flows to guide users through IAM activities.

A flow is a set of paths that define the user journey through a given IAM process, such as registration or authentication, using a set of logically linked nodes. Each node in a flow either takes an action on the backend or presents the user with a page. The path a user follows through a flow is determined by logical decision operators that sit between the nodes, enabling you to control how a user progresses based on the information they provide, existing user information, or other parameters.

Each node performs a single task in the flow. When you place a node, you associate it with a connector and then select a capability provided by that connector. These capabilities include retrieving information from third parties, presenting customized pages to users, modifying the value of variables, or performing other actions.

You can add new connectors to your environment to make new capabilities available in your flows. These connectors can communicate with other Ping products or third parties. You can also configure multiple instances of a connector to use different environments or settings. The connector documentation and the integration directory contain information about the available connectors and the capabilities they provide.

After you create a flow, you add it to an application and create a flow policy to control how and when the flow gets used. Applications enable you to run flows using a widget, API calls, OpenID Connect (OIDC) calls, or Security Assertion Markup Language (SAML) 2.0 calls. The application also lets you perform A/B testing by splitting traffic between different flows or versions of the same flow.

DaVinci includes tools to enable customization and information tracking in your flows. The UI Studio lets you configure HTML pages that match your company's style and branding, which you can then present to users within flows. Variables let you track data within flows or across multiple flows and use that data to determine a user's path.

Use this documentation for the core functions and usage of DaVinci. For details about individual connectors, see the integration directory.

Flows

A flow is a user journey, such as registration or authentication, built from a set of capabilities and logical operators.

Structure

Every flow consists of one or more nodes joined together by logical operators. Each node performs a specific task, using one of the capabilities of your connectors. After the task is complete, the logical operators determine which task or tasks are performed next. The user begins with the leftmost node and moves to the right along all applicable paths, unless you create a path that loops them back to an earlier step.

A flow can have multiple paths, and depending on the logical operator, more than one branch can be active at one time. For example, if your initial node has two "True" logical operators leading from it to a create token node and a send email node, the two nodes can run at the same time. In each case, the flow progresses to the right until the branch ends or until an error occurs.

Nodes

Each node in a flow represents a specific task. When you create a node, you select a connector and a capability that connector supports. For example, a node using a Google connector gives you the capability to sign on a user with Google credentials, and a node using an HTTP connector gives you the capability to display sign on pages or forms to the user.

Nodes with a dotted border indicate that the node has some amount of user interaction, such as a sign on form or a status display. Nodes with a solid border are performed without displaying anything to the user.

Logical operators

Nodes are joined together by logical operators. One or more nodes can feed into a logical operator, but each logical operator leads to exactly one node. The logical operators determine which node or nodes are run next:

Logical Operator	Description
	If all of the nodes leading to this operator completed successfully, the subsequent node runs.
	If any of the nodes leading to this operator completed successfully, the subsequent node runs.
	If all of the nodes leading to this operator failed to complete successfully, the subsequent node runs.
	If any of the nodes leading to this operator failed to complete successfully, the subsequent node runs.
	If all of the nodes leading to this operator completed successfully or unsuccessfully, the subsequent node runs.
	If any of the nodes leading to this operator completed successfully or unsuccessfully, the subsequent node runs.

How to create a flow

To create a new flow in DaVinci, you add nodes that perform specific actions and join them together with logical operators.

Creating the flow

To begin creating a new flow, click **Add Flow** on the **Flows** tab. You'll be presented with three options:

- Template Flow: Create a flow starting with a template created by Ping.
- Import Flow: Import a flow that was exported in JavaScript Object Notation (JSON) format. If you import a flow that uses a connector that you haven't configured, you'll need to configure it before using the flow.
- Blank Flow: Create a flow starting from a blank canvas.

Adding nodes

Nodes represent specific actions within a flow. When you first create a flow, it has no nodes, but you can add nodes in multiple ways:

- Click the **+** icon in the lower left of the flow canvas. This creates a new node without any connection to an existing node.
- Click and drag from the start or stop icons on the left or right of an existing node. This creates a new node before or after the existing node.
- Click and drag from the dot on the left or right of an existing logical operator. This creates a new node before or after the logical operator.
- Right-click an existing node and click **Clone**.
- Select one or more existing nodes, right-click them and click **Copy**, then right-click the canvas and click **Paste Nodes**.

Using logical operators

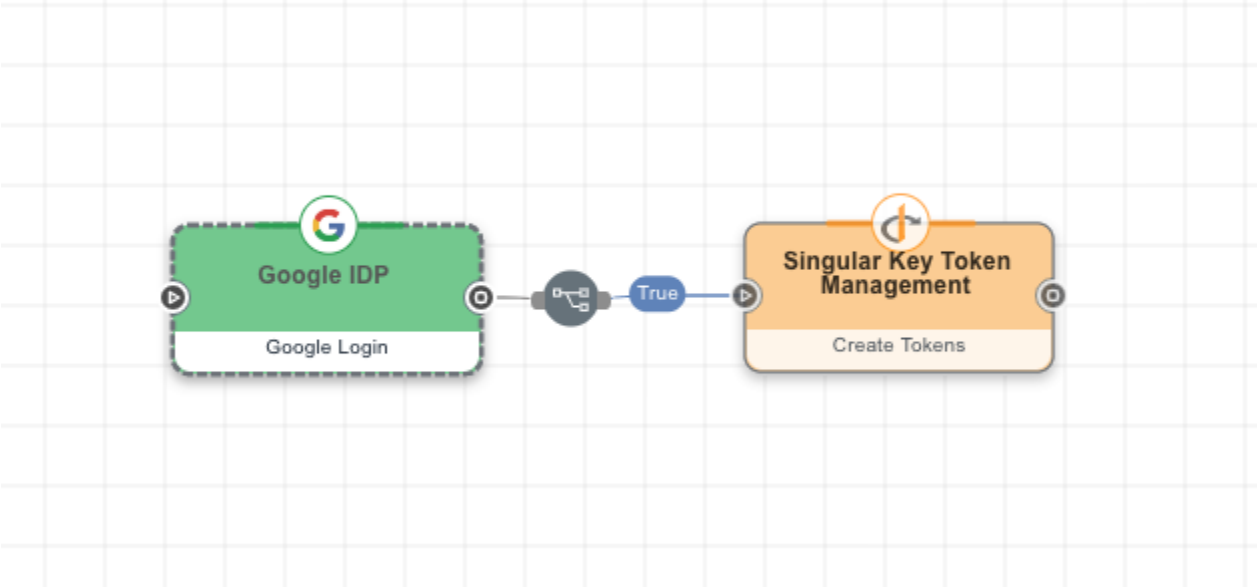
When you click and drag a line from one node to another, a logical operator is created between the two nodes. This logical operator determines whether the subsequent node runs.

A label indicates the type of logical operator. To change the type, right-click the label or the line itself, then select another type.

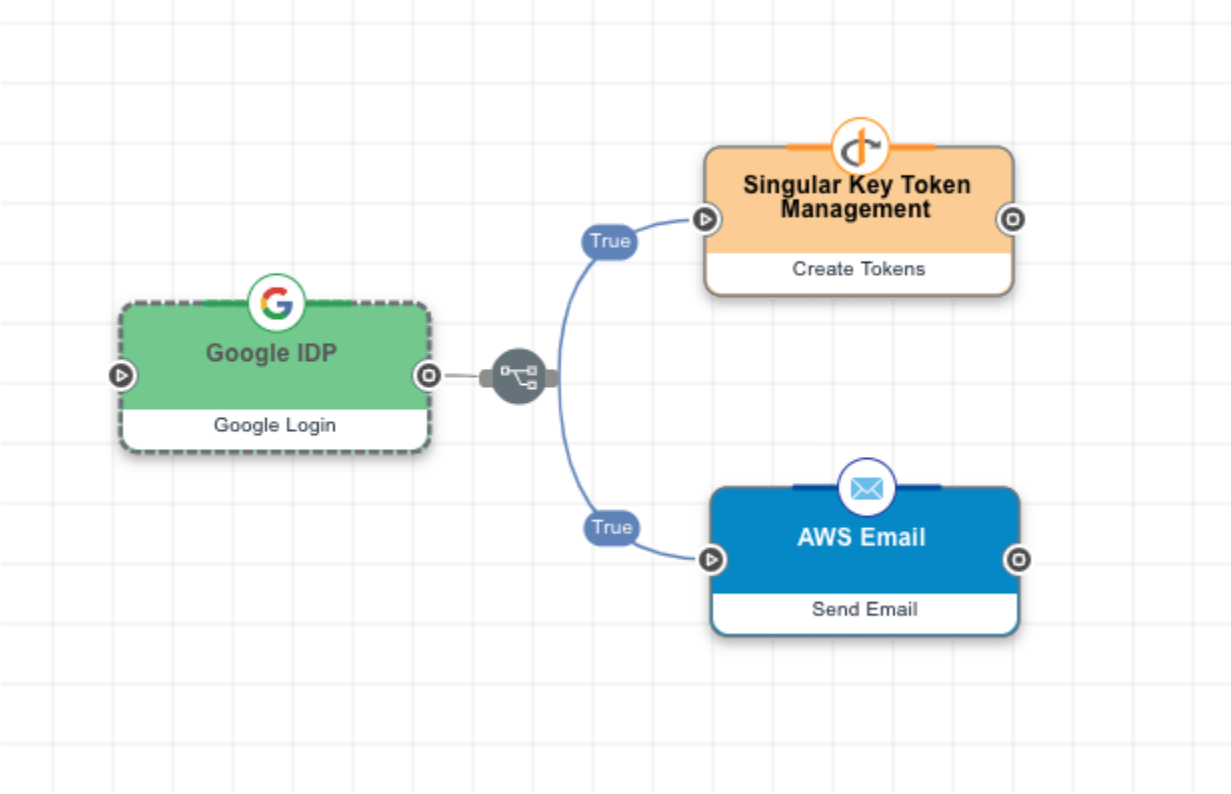
Flow patterns

By connecting nodes with logical operators, you can create relationships between the nodes.

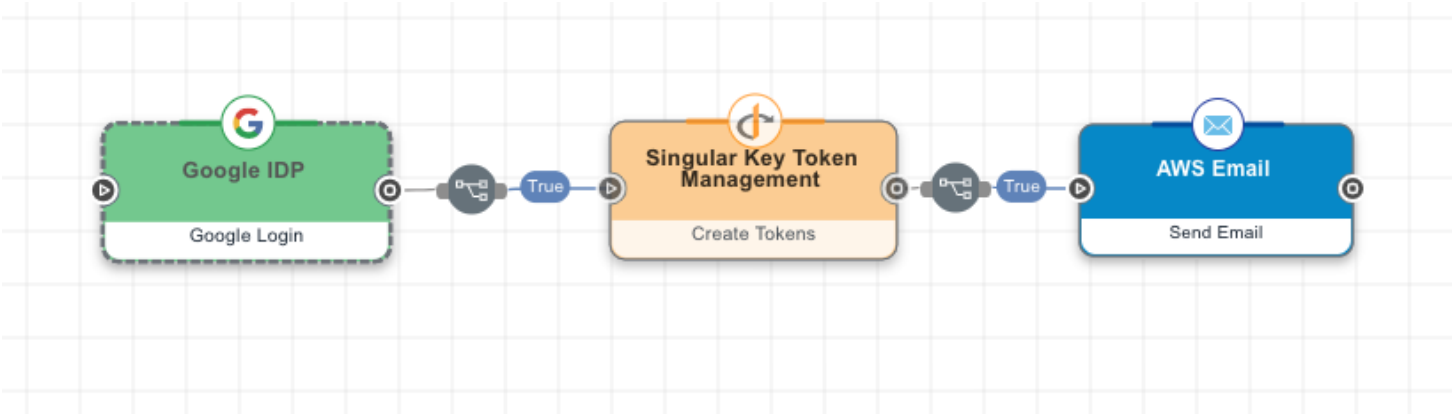
In this example flow, the user is first prompted to sign on with Google. After the user successfully authenticates, the second node creates tokens for the user and redirects them back to the relying party.



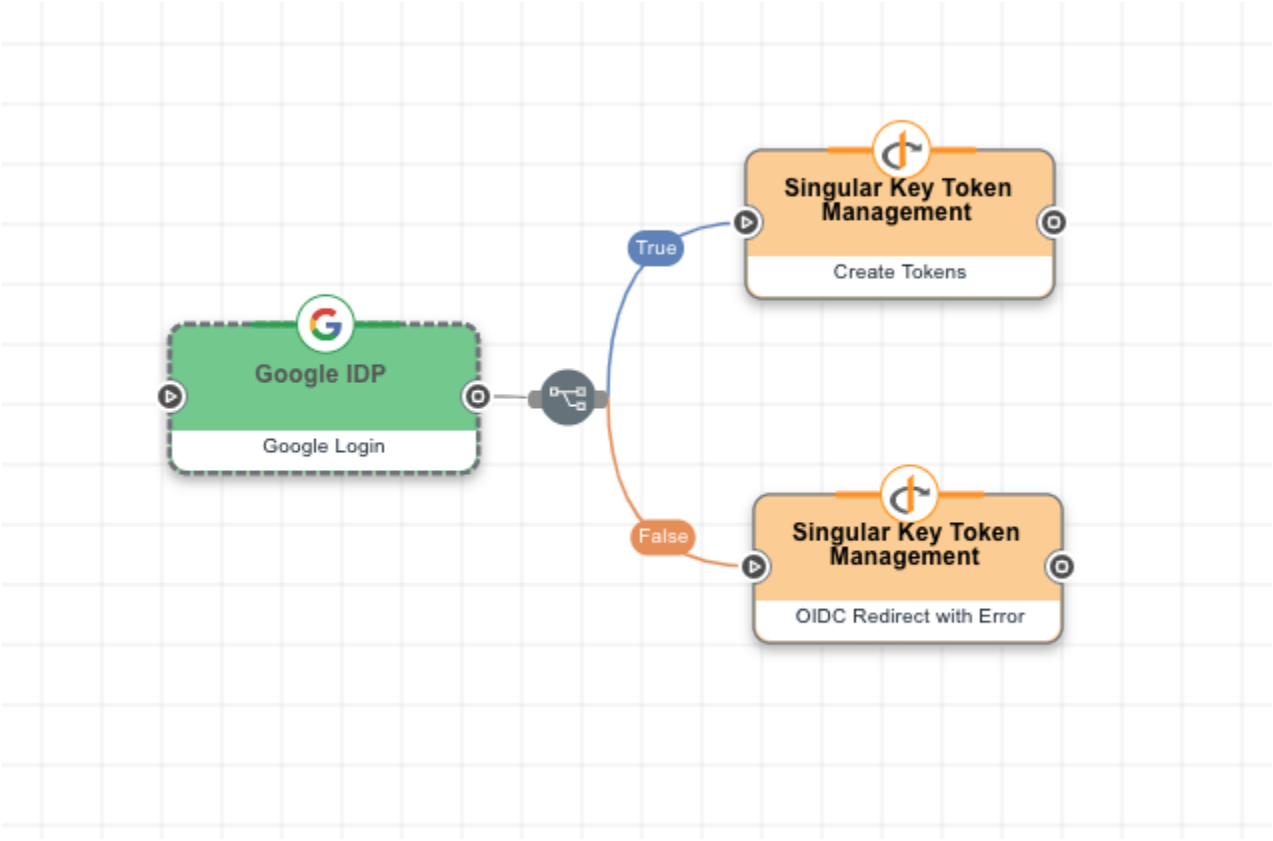
One node can trigger more than one additional node when it completes. In this example, after Google login completes, the flow creates tokens for the user and sends an email to the user.



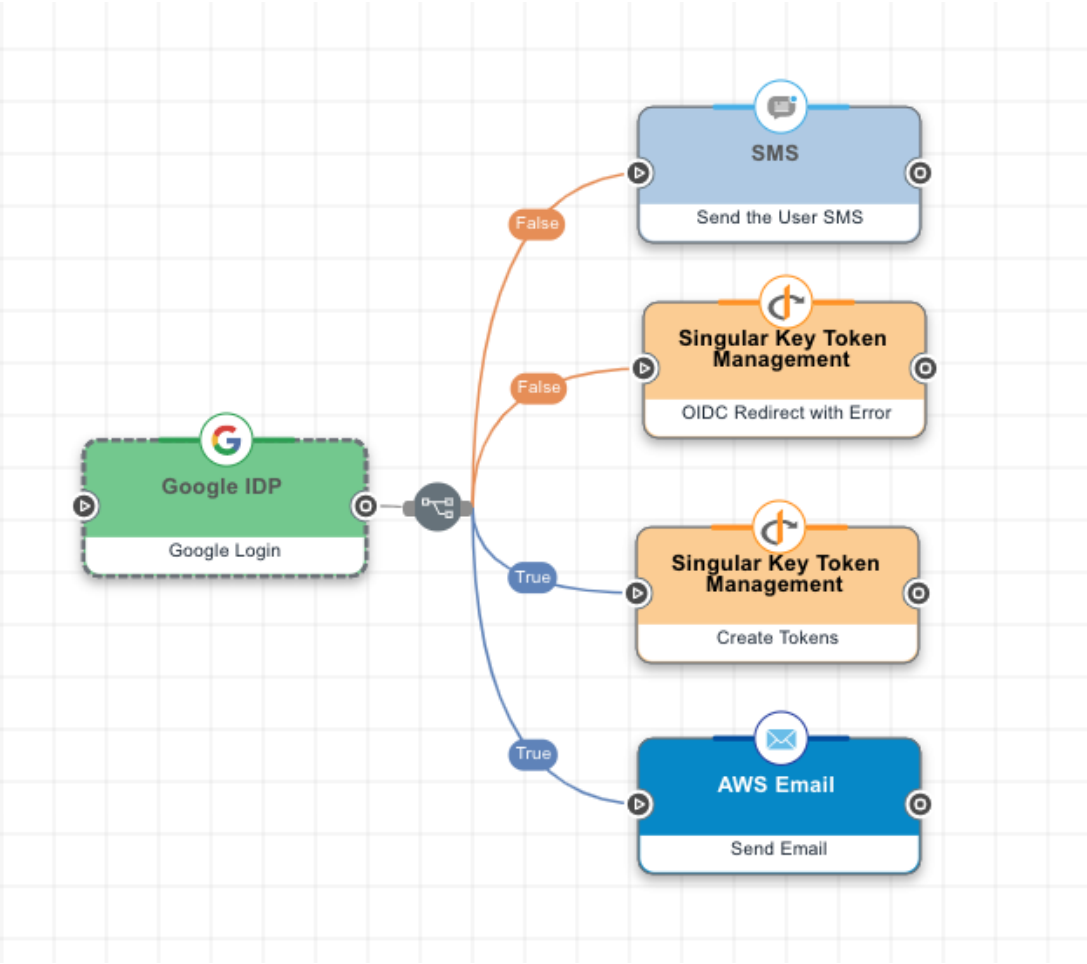
This example uses the same nodes as the previous example, but the email is only sent after the tokens have been successfully created.



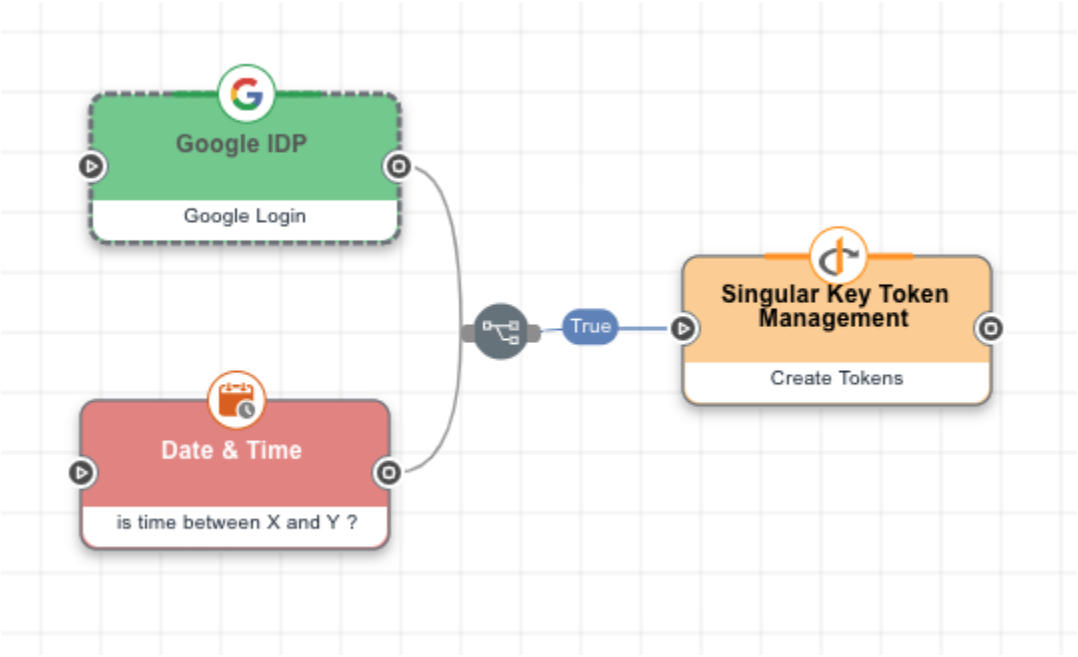
This example flow deals with a scenario where the Google IdP login fails. If it succeeds, the tokens are created, but if it fails, an error message is displayed.



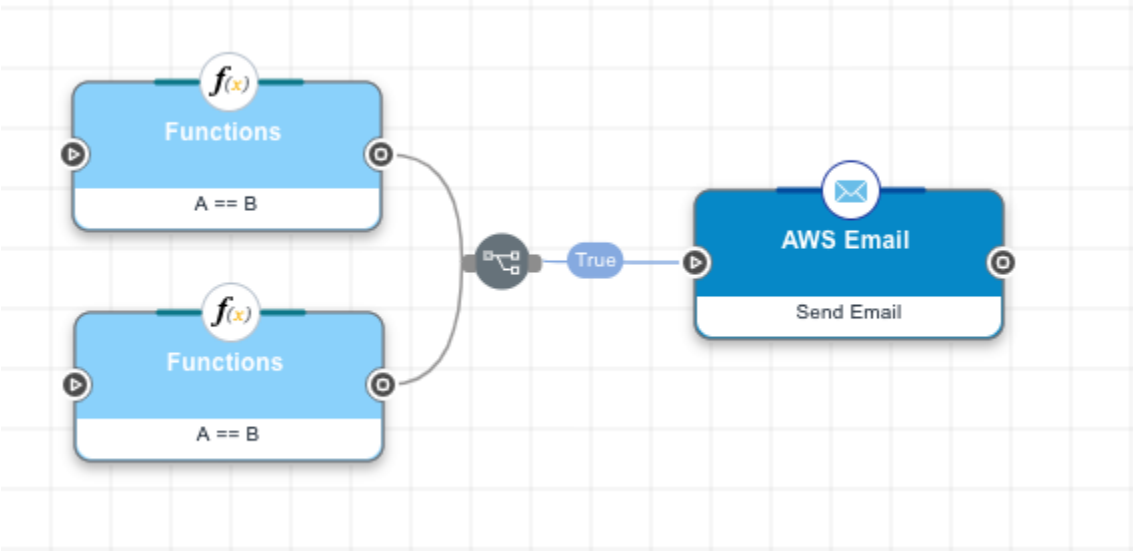
A single node can trigger multiple nodes, if it fails or succeeds. In this example, two nodes are launched if Google authentication succeeds, and two other nodes are launched if it fails.



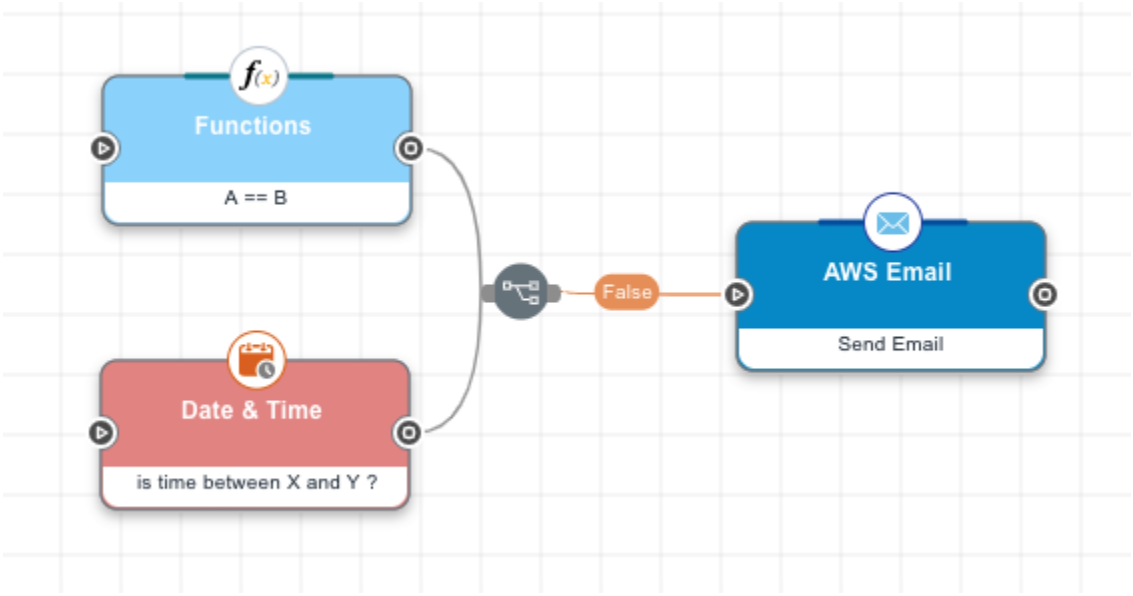
A node can have more than one trigger. In this case, the two trigger nodes have an AND relationship because the logical operator is "If All True". This means that the tokens are only created if the Google login succeeds and if the date and time are within the specified range.



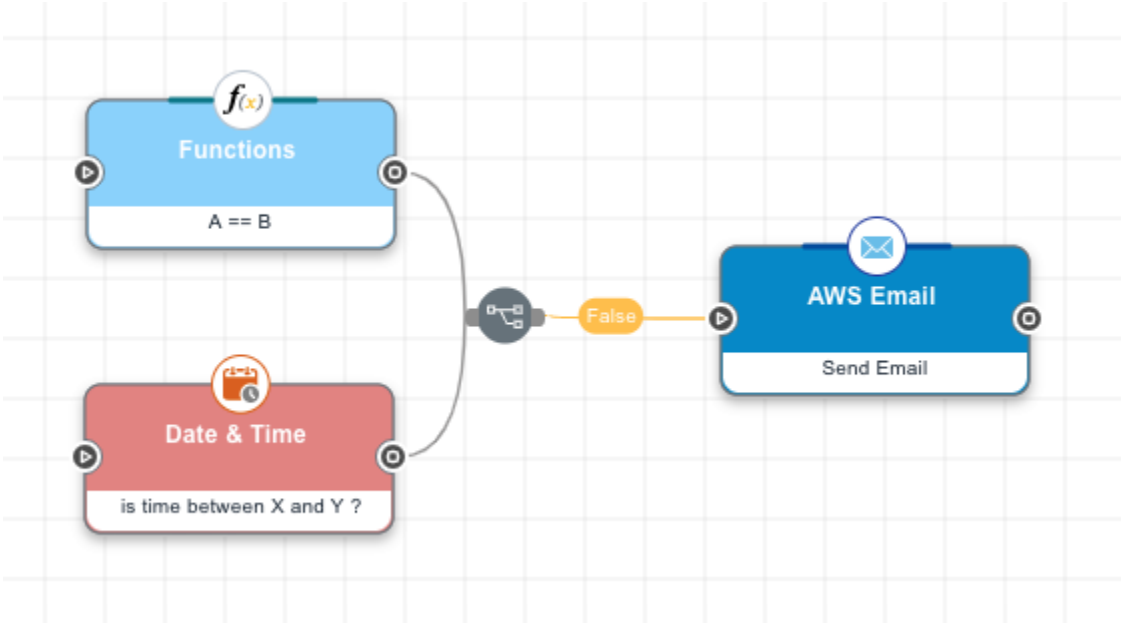
In this example, the two trigger nodes are linked to the Amazon Web Services (AWS) Email node by an "If Any True" logical operator, meaning that the email node is triggered if either of the initial nodes completes successfully.



In this example, the AWS Email node is triggered if both of the preceding nodes complete unsuccessfully.

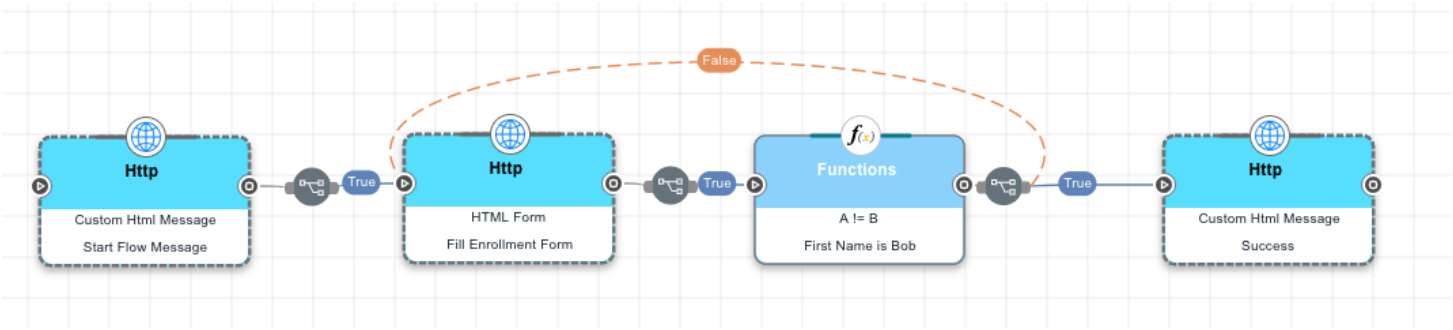


In this example, the AWS Email node is triggered if either of the preceding nodes completes unsuccessfully.



You can loop a node back to a previous node by clicking its output and drawing a line back to the earlier node. In this example, a Functions node checks the information provided on an HTML form and sends the user back to the form if the data doesn't fall within the specified parameters.

> **Note**
>
> You can't loop back to the first node in the flow.



Use cases

Flows are, by design, a broad toolset. This section explains the principles of assembling a flow. For examples of flows designed for specific scenarios, see the Use Cases section.

Getting started with DaVinci

Let's explore how to create your first flow.

Choosing a use case

You're here for a reason. There's a customer experience that you want to create or improve without having to write a mountain of code. DaVinci is a tool to help you do that.

Start your process by identifying the problem that you want to solve. Usually identity experiences start by focusing on sign on and registration. But there's a lot more you'll want to eventually consider, such as what happens when a user forgets their username or password or needs to set up multi-factor authentication (MFA).

If you already have the basics in place, you might be looking for ways to streamline the end user experience without increasing the risk of inappropriate access, such as incorporating data from device management, risk scoring, or behavior analytics services. If you're looking for inspiration, check out the flow templates in the Integration Directory.

Another key area you might be focusing on is giving your end users new capabilities, such as applying for a credit card, adding information to their profile, or setting up a new authorized user. These self-service activities can be quickly prototyped and integrated into existing web applications.

Don't forget about support flows either, which often start from a chat box or phone call but greatly benefit from authentication to verify a user's identity and link it to their official account information.

No matter what use case you are focusing on, before you even open up DaVinci, grab a pen or whiteboard marker and sketch out the logic of the experience you want to create. Now let's start building!

Creating the flow

After you have a good understanding of what you want your flow to do, you can begin putting it together.

There are a few ways you can create a flow. Ping Identity has created a variety of flow templates that cover many common use cases. If the flow you're imagining is similar to one of these use cases or incorporates parts of one, you can use the template as a starting point.

You can download these templates as JavaScript Object Notation (JSON) files and import them into your environment. For help, see Using DaVinci flow templates.

If you have an existing flow that you want to use as a foundation, you can import the JSON for that flow or clone it to create your new flow. To import the JSON, click **Add Flow**, select **Import From JSON**, name the flow, and click **Import**. To clone a flow, select the existing flow and click **... → Clone**.

If your flow concept is different from existing templates or flows, you might need to start from an empty flow canvas. To create a blank flow, click **Add Flow**, select **Blank Flow**, name the flow, and click **Create**. You can then build out the flow to fit your specific use case.

Starting with a key part of the experience

After you have started your DaVinci flow, there are a couple of ways that you can progress.

If you started with a template or an existing flow, you can begin by testing how the flow currently works. You can then make changes to the structure to bring it in line with what you're imagining.

If you're starting from a blank slate, the process is more open. A good way to begin is to prototype the front-end experience first. You can use the **HTML Form** capability of an HTTP connector to represent user-facing pages and the **Custom HTML Message** capability of another HTTP connector to represent messages displayed to the user. Use these capabilities to build a prototype and figure out how you want users to progress. Test the user experience to make sure it matches what you imagine.

Using the **HTML Form** capability of the HTTP connector is a great way to prototype user inputs without having to write any HTML code. For messages displayed to the user, the **Custom HTML Message** capability provides a quick way to print out data. As you get closer to making your flow pixel perfect, you can substitute in the **Custom HTML Template** to give you full control over the HTML, CSS, and JavaScript.

After you have the user-facing piece assembled, you can add nodes to perform backend actions and replace the test HTML forms with pages that match your style and branding.

To map out the logical execution of your flow, start by using the Functions connector. This allows the logical evaluation of outputs from previous nodes to determine forks in the flow. To make your flow more functional, you can add REST API calls or connectors that can interact with external systems.

Dropping our first node

Your flow is built out of multiple nodes, with each node using a single capability from one of your connectors.

You can add a node to the flow canvas in multiple ways. You can click the **+** icon to add a node without any logical links to existing nodes. You can also click and drag a line from an existing node, and a new node is automatically added.

When you create a new node, you specify which connector it uses. If the connector you want isn't on the list, you can add it on the **Connectors** tab by clicking **Add Connector**.

Many connectors have multiple capabilities. For example, a PingOne connector lets you look up users or create new users. When you create a node, you select one capability. The node will show you what information is required for that capability.

You can do a lot of additional configuration to the nodes in your flows. You can name them to make their purpose clear, change their background colors to indicate their role in the flow, and provide additional information, such as logging details.

When you're done updating a node, click **Apply** to apply your changes to the node.

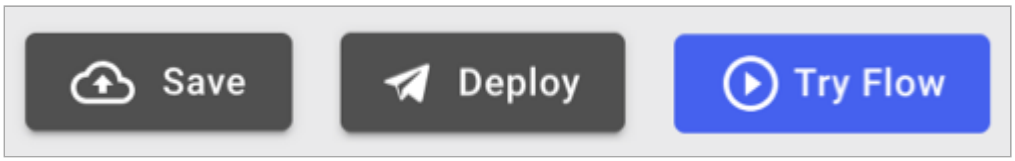
> **Important**
>
> Changes to the node aren't preserved if you don't apply them.

You can copy and paste individual nodes or groups of nodes into the current flow or into a different flow. Select the nodes, right-click, and select **Copy**. You can use this to reuse components of existing flows.

Testing early and often

Testing your flows frequently is key to making them work correctly.

At the top of the flow canvas are three buttons: **Save**, **Deploy**, and **Try Flow**.



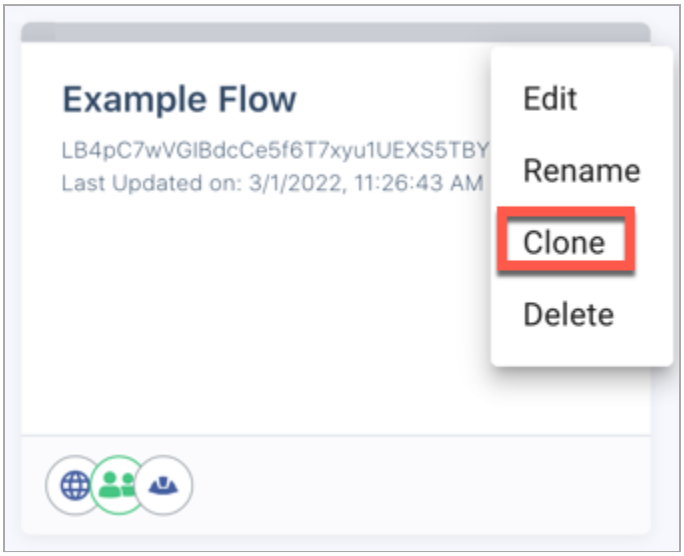
Save saves all changes to the flow and creates a new version in the version list. **Deploy** deploys the current version of the flow. **Try Flow** launches a test run of the flow.

When you're developing a flow, use these options regularly. Save the current version of the flow, deploy it, and then try it to make sure it's behaving the way you expect.

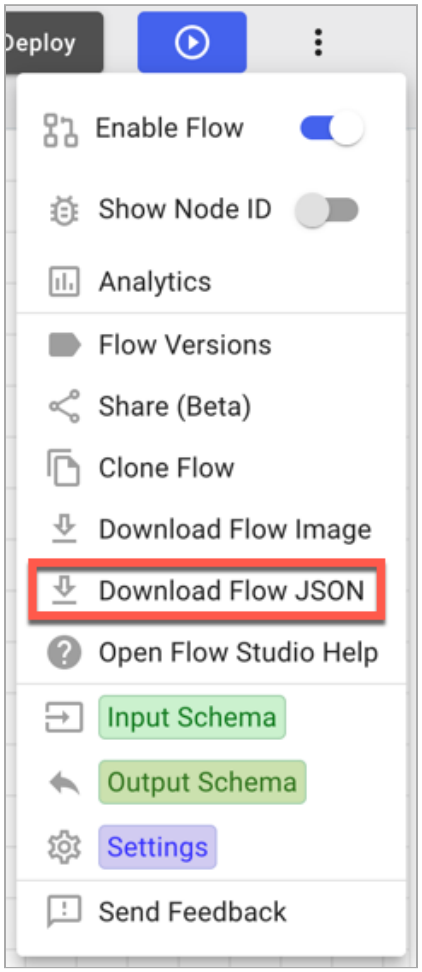
If you encounter a new issue with your flow, you can revert to a previous version to prevent the issue and try to diagnose it. Click ⋮ → **Flow Versions**, select the earlier version in the list, and click **Revert**.

You can also disable one or more nodes as part of your troubleshooting. Select and right-click the nodes, then click **Disable**. This prevents the disabled nodes from being activated during an execution of the flow.

If you want to make a modified version of a flow or use one flow as a basis for a new flow, you can clone or export an existing flow. Cloning a flow creates an identical but distinct copy of the flow with a timestamp of when it was cloned. To clone a flow, locate the flow and click **...** → **Clone**.



To export a flow, open the flow, then click ⋮ → **Download Flow JSON**. When you create a new flow, you can choose to import the saved JSON file as a starting point, even in a different environment.



Troubleshooting your flow

A vital part of building a successful flow is troubleshooting errors when things don't go as expected. The following are suggestions and techniques to assist in getting your flow back on track.

Clicking Try Flow doesn't launch flow

If **Try Flow** is grayed out, verify that your flow has been deployed. After you save your changes, click **Deploy** to publish them. Additionally, verify that the flow contains UI components.

Flow will not deploy

If a flow does not deploy, verify that it does not include nodes that are missing connector configuration information. Verify that all of the nodes in the flow are connected to the preceding and subsequent nodes.

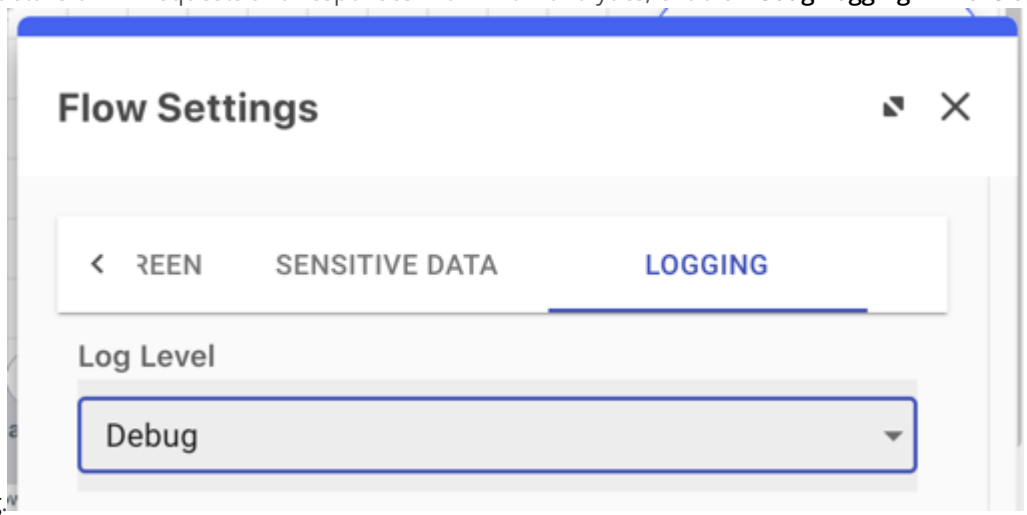
Flow fails to continue

If a flow progression halts, most likely it's waiting on user interaction. Adding a **Continue** button to the form will allow the flow to progress to the next event.

Another possibility is that your flow encountered an error when processing. In the flow, click **Analytics** to display a list of flow executions. Select the most recent flow run to view the path and details of each node.

Enable Flow debug logging

To view additional details of API requests and responses within flow analytics, enable **Debug Logging** in **More options (⋮) → Flow Settings → Logging.**



Temporary UI screens

To view the current state of variables, API responses, and other flow data, place temporary UI components within the flow to display their details.

Node IDs

As your flow grows, you might find it helpful to display the node IDs. This allows you to verify that values are being pulled from the correct node. To see where a value such as `{{email}}` is pulled from, mouse-over the name to display its details (for example, `{{local.m3qhb63tfb.payload.output.email}}`). In this example, `m3qhb63tfb` is the node id that contains the email attribute.

Flow execution paths

If your flow doesn't progress correctly, verify the logical operators between nodes to ensure the right conditions are being met (All triggers true, Any trigger false, and so on.).

Invoking your flow

After you've created a flow and performed early testing, you can launch the flow.

Flows can be launched in multiple ways, depending on the content of the flow and the way you want it to appear. See the developer documentation for more information about launching flows using these methods.

Invoking a flow using a redirect

You can launch a flow using a whole-page redirect. The user is sent to a new page with your DaVinci domain in the URL. This method works well for flows involving user interaction in situations where you don't mind redirecting the user to a new page.

Invoking a flow using the widget

You can launch a flow in a widget on an existing page. The user stays on the same page, and the flow is displayed in a portion of the page. This method works well for flows involving user interaction in situations where you want the user to remain on your page.

Invoking a flow using an API call

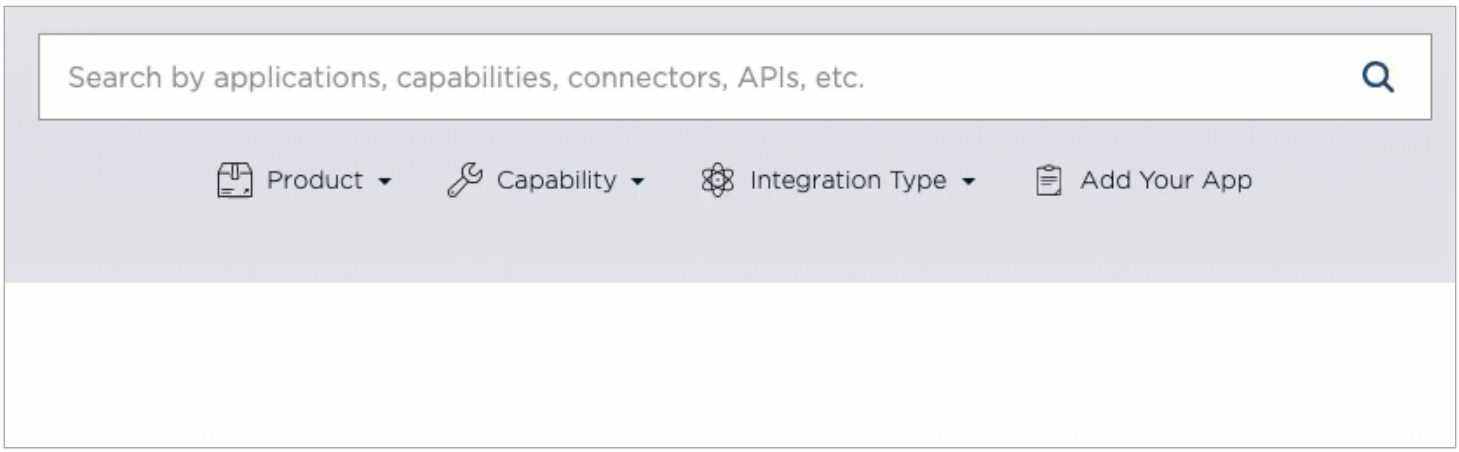
You can launch a flow using an API call. This method works well for flows without any direct user interaction.

Using DaVinci flow templates

DaVinci flow templates offer premade solutions for common orchestration use cases. You can modify them to meet your needs or use them as inspiration when designing your own flows.

Finding flow templates

The Ping Identity Integration Directory allows you to use filters and search terms to find DaVinci flow templates for specific use cases. Click a listing to see more information about a flow template. You can also view these templates when you create a new flow using the **Template Flow** option.



Adding flows using flow templates

1. In DaVinci, on the **Flows** tab, click **Add Flow**.
2. Click **Template Flow**.
3. Find the template you want to use and click **Use Template**.

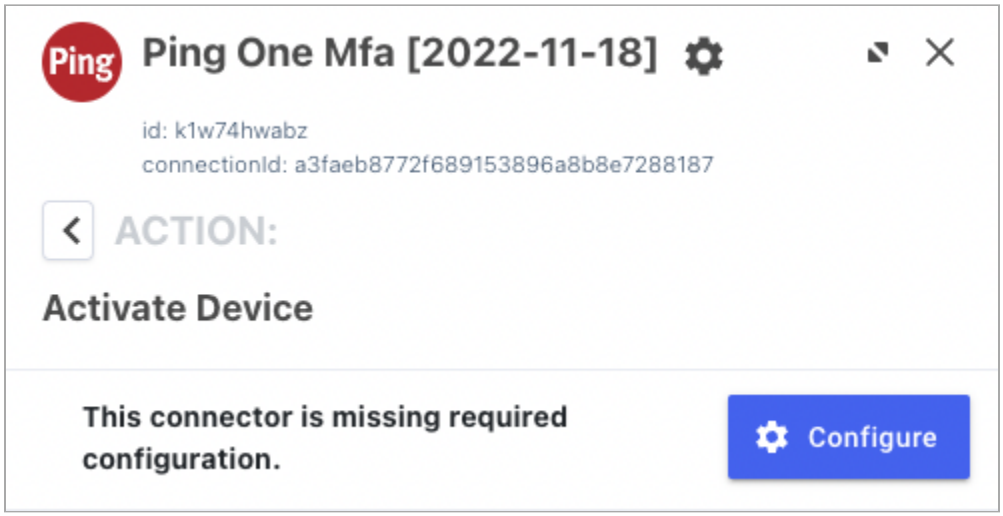
   You can click **Learn More** to view additional information about the template.
4. Enter a name and description for your flow. Click **Add**.

Configuring new connectors

When DaVinci creates a new flow from a template, it uses your existing connectors. If you have multiple instances of a connector, DaVinci uses the oldest connector instance in the environment.

If the flow needs a connector that you haven't created before, DaVinci automatically creates the connector instance and gives it a name, such as **PingOne MFA [2022-12-20]**.

To use the flow, configure these new connectors. You can configure a connector by selecting it on the flow canvas and clicking **Configure** in the details pane.

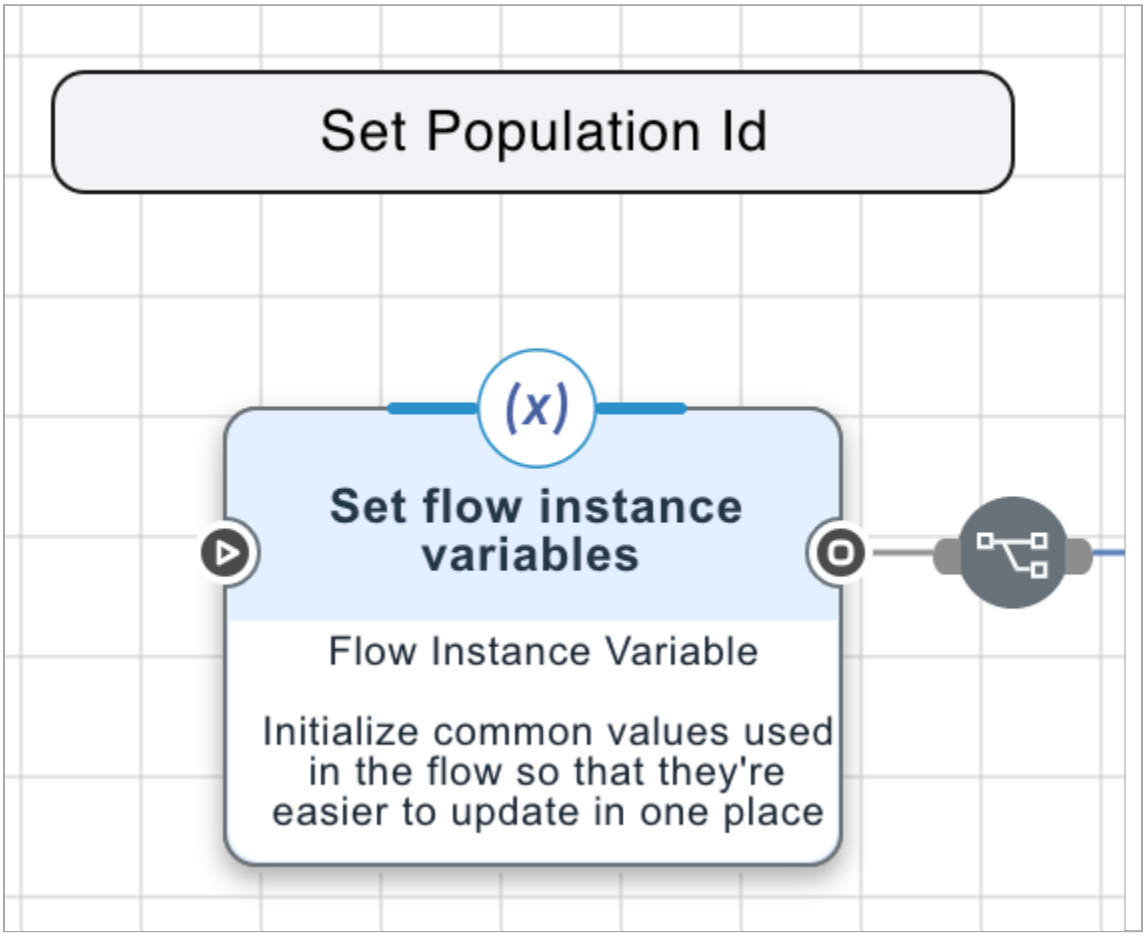


For help with the connectors configuration, check the connector documentation. You can find links to the connector documentation in the Integration Directory listing where you downloaded the flow template.

Configuring variables for Ping-built flows

Ping Identity's flows collect variables in a single node to make it easy to configure the flow.

To configure the variables, click the **Variables** node at the beginning of the flow.



Learn more about configuring specific variables in Common flow variables.

Integrating flows into your environment

After you finish configuring and testing your flow, you can integrate it into your organization's test or production environment using the widget, redirect, or API method. Learn more in Integrating flows into your environment.

Common flow variables

Flow templates created by Ping Identity use variables to make it easy for you to configure the flow. Although each flow uses a unique set of variables, the most common ones are defined in this list.

> **Note**
>
> If your flow uses variables that aren't listed below, check the flow template listing in the Integration Directory for help.

To configure the variables, go to the **Variables** tab in the main DaVinci menu. If a flow uses variables that you already have set, you don't need to configure them again. If a flow uses new variables, DaVinci automatically adds them to your **Variables** list.

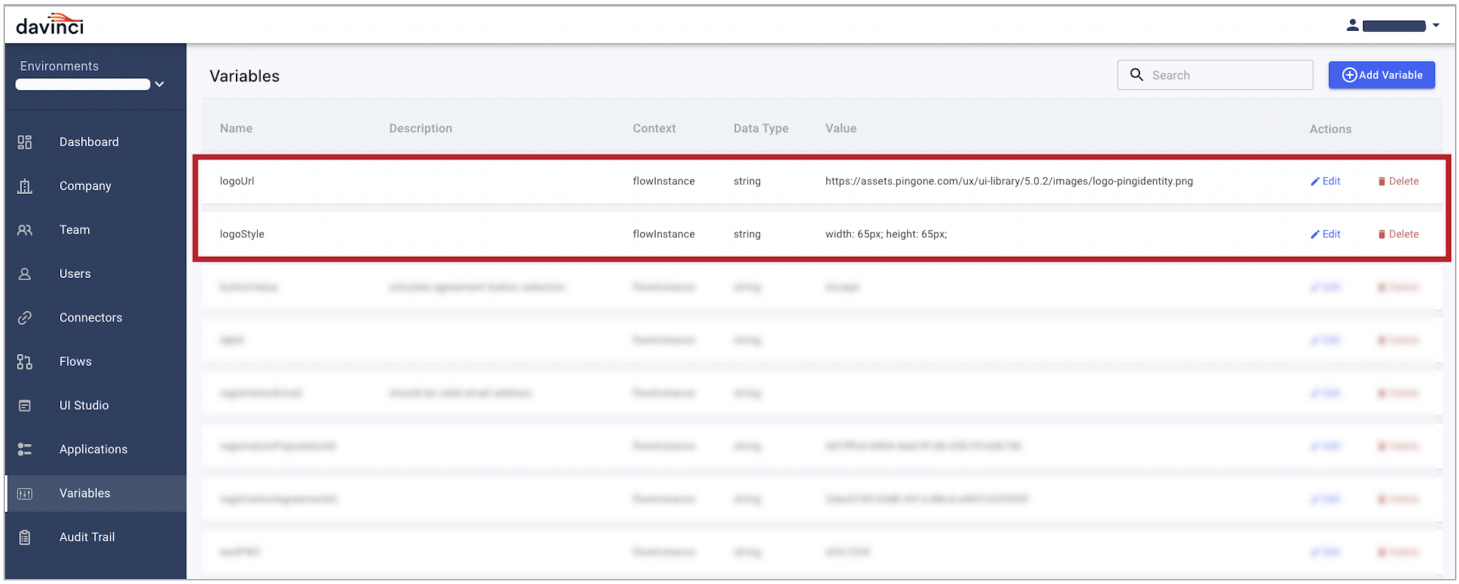


Table 1. Common Flow Variables

Variable Name	Description
origin	Your PingOne domain, such as `auth.pingone.com`. **Note** Change this value if your region is Canada, EMEA, or APAC, or you have a custom PingOne domain.
originURL	Your PingOne URL, such as `https://auth.pingone.com`. **Note** Change this value if your region is Canada, EMEA, or APAC, or you have a custom PingOne domain.
companyLogo	Your company logo image URL, such as `https://cloudacme.example.com/logo.png`. This enhances the user experience with your brand.
companyName	Your company name. This enhances the user experience with your brand.
relyingParty	The name of the relying party, such as `PingOne`.
relyingPartyID	The ID of the relying party, such as `auth.pingone.com`. **Note** Change this value if your region is Canada, EMEA, or APAC, or you have a custom PingOne domain.

How to manage flows

After you create a flow, you can take actions to manage it.

- Flow editing options
- Managing nodes
- SK-Components reference
- Viewing flow analytics
- Using load balancing
- Using flow search
- Editing the input schema
- Editing the output schema
- Editing flow settings
- Adding annotations
- Validating a flow
- Renaming a flow
- Cloning a flow
- Exporting a flow
- Deleting a flow
- Flow limits

Flow editing options

You can control many flow properties using the **More options (⋮)** menu while editing a flow.

Option	Description
More options (⋮) → Enable Flow	Enables or disables the flow. Disabled flows cannot be used by end users.
More options (⋮) → Show Node ID	Displays the node ID for each node, just beneath the node itself.
More options (⋮) → Analytics	Displays flow analytics. For more information, see Viewing flow analytics.

Option	Description
More options (⋮) → Flow Versions	Displays a list of versions of the flow. This list notes the latest version, the deployed version, and the changes that were made in each version.
More options (⋮) → Clone Flow	Clones the flow. For more information, see Cloning a flow.
More options (⋮) → Download Flow Image	Downloads a PNG image of the flow.
More options (⋮) → Download Flow JSON	Downloads a JavaScript Object Notation (JSON) file containing the complete flow.
More options (⋮) → Input Schema	Configures the flow's input schema. For more information, see Editing the input schema.
More options (⋮) → Output Schema	Configures the flow's output schema. For more information, see Editing the output schema.
More options (⋮) → Flow Settings	Configures the flow settings. For more information, see Editing flow settings.

Managing nodes

Manage the nodes included in your flow.

- Node properties
- Cloning a node
- Copying nodes
- Replacing a node
- Changing a connector instance
- Disabling or enabling a node
- Deleting a node

Node properties

You can edit the properties of any node in your flow.

General and connector-specific tabs

The **General** tab contains properties related to the selected connector and capability. See the connector's entry in the integration directory for information about these properties.

Some connectors and capabilities use additional tabs. These tabs are unique to the connector and are described in the connector documentation.

Settings tab

The **Settings** tab contains settings about the node display and performance. After you edit these settings, click **Apply** to save your changes.

Property	Description
Node Title	The node title displayed in the user interface. This title is not visible to end users.
Node Description	An internal description of the node. This description is not visible to end users.
Node Background Color	The background color of the node in the user interface. This color is not displayed to end users.
Expire Authentication Token	Causes the access token or SDK token to expire when the node completes.
Expire Flow Instance Cache	Removes cached information about the flow when the current node completes.
Expire Node Instance Cache	Removes cached information about one or more nodes when the current node completes.
Expire Node Instance Cache List	The Node IDs for the node or nodes to remove from the cache if **Expire Node Instance Cache** is selected. Press `Enter` after entering each Node ID.

Schema tab

The **Schema** tab lets you view and test the node schema. When the tab is opened, the **Default** view is displayed. It uses a color-coded view to display details of the schema, such as data types, required parameters, and parameter constraints.

Property	Description
Default	Displays the default view of the schema.
Raw	Displays the raw schema.
Example	Displays an example of the schema.

Mappings (Error Messages) tab

The **Mappings (Error Messages)** tab specifies additional information to add to error logs for the given node.

Property	Description
+	Adds an additional key-value entry to the table.

Property	Description
Key	A key name to include in error logging.
Value	A value to include in error logging, corresponding to the **Key** field in the same row. Click **\{}** to select a flow property, data provided in a previous node, or variable.

Log Fields Mapping tab

The **Log Fields Mapping** tab lets you map flow properties to variables in the logs.

Property	Description
Edit	Allows you to delete one or more mappings. Click **-** to delete a mapping and **Close** to return to the standard view.
+ Field	Adds an additional mapping.
Global Variable	The global variable.
Mapping Variable Name	The value to map to the global variable.

Cloning a node

Clone a node to add an identical node to the flow.

Steps

1. Click the **Flows** tab.
2. Locate and open the flow.
3. Right-click the node.
4. Click **Clone**.

Copying nodes

Copy one or more nodes to paste them into the current flow or into another flow.

Steps

1. Click the **Flows** tab.
2. Locate and open the flow.
3. Select one or more nodes.
4. Right-click the nodes.

5. Click **Copy**.

Result

The nodes are copied. Paste the nodes by right-clicking the background of any flow and clicking **Paste Nodes**.

Replacing a node

Replace a node to create a new node in its place.

About this task

> **Note**
>
> Replacing a node that is already configured will remove any configurations, such as a selected capability and variable mappings. When replacing a node with another node of the same type (by changing the connector instance), you can preserve configuration information between them. For more information, see Changing a connector instance.

Steps

1. Click the **Flows** tab.
2. Locate and open the flow.
3. Right-click the node.
4. Click **Replace**.

   The **Replace Connector** window displays.
5. Select a different connector.

   Choose from:

   - Select a connector from the **Existing** tab to immediately replace the connector in your flow with a connector already added to your environment.
   - Select a connector from the **New** tab to add a new connector to your environment.
   - Give the new connector a name and click **Create**.

Next steps

Configure the new node according to the documentation for the selected connector.

> **Note**
>
> If you selected from the **New** tab, open the connector and click the **Edit Connector** icon to configure the connector for your environment.
>
> 
>

Changing a connector instance

To use a different instance of a selected connector while retaining its in-flow configuration, change the linked connector.

About this task

> **Note**
>
> When you change the linked connector for a node, all other nodes in the flow that use the same connector type are updated to use the new linked connector.

Steps

1. Click the **Flows** tab.
2. Locate and open the flow.
3. Right-click the node.
4. Click **Change Linked Connector (for nodes of the same type)**.
5. Select a new connector instance from the **Replace Connector** window.

Result

All nodes in the flow of the selected connector type now use the new connector instance.

Disabling or enabling a node

Disable a node to temporarily prevent it from being executed when the flow is run. Enable it to execute the node as normal.

Steps

1. Click the **Flows** tab.
2. Locate and open the flow.
3. Right-click the node.
4. Click **Disable** or **Enable**.

Deleting a node

Delete a node to remove it from the flow and break any connections to it.

Steps

1. Click the **Flows** tab.
2. Locate and open the flow.
3. Right-click the node.
4. Click **Delete**.

A confirmation modal opens.

5. Click **Delete**.

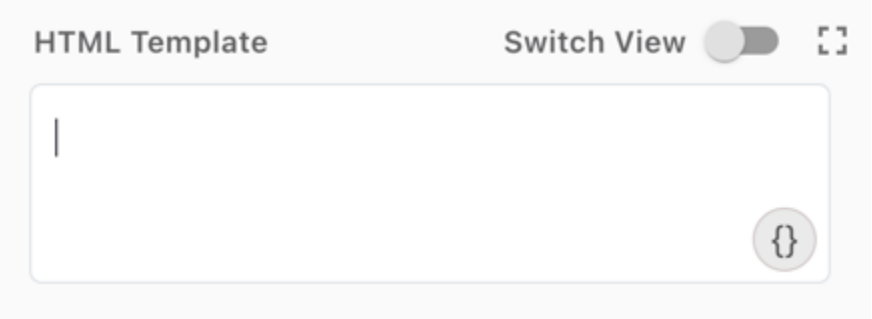
Including variables and other data

Many of the nodes in a flow can make use of data from outside of the node. This data can include:

- **Global variables**: Parameters available throughout the flow. These parameters can come from the flow invocation, the environment, or other sources.
- **Node-specific data**: Many nodes create or import data, such as user details, which is then available to future nodes.
- **Variable values**: You can include variable values from relevant variable types.

To include one or more of these parameters in a node:

1. Create or open a flow.
2. In your flow, select a node, then locate a field that requires data.
3. In the field, click the **{}** icon.



4. In the **Choose Connection** list, select **Global** to add a global variable, select a previous node name to add data from that node, or select a variable type to add a variable value.

   The selected category expands to display specific

> **Note**
>
> By default, you can only select nodes that precede the current node. If the flow contains loops, you can click **Show All Nodes** to display nodes that appear later in the flow.

5. Select a value to include.
6. Click **Apply** after you finish adding parameters.

Global variable reference

Global parameters are parameters available throughout the flow.

Parameter	Description
`companyId`	The company ID for the environment.
`flowId`	The ID of the flow.
`flowVersionId`	The flow version number being run.
`interactionId`	The ID for the flow execution.
`policyId`	The ID of the flow policy used to launch the flow.
`ip`	The end user's IP address.
`userAgent`	The end user's user agent.
`skOpenId`	An object containing authorization request parameters.
`cookies`	The cookies present in the API request that invoked the flow.
`origin`	The value of the Origin header.
`originCookies`	A string used to address the case where the UI the user is interacting with has a different domain than the domain the widget is making API calls to.
`xDeviceTrust`	A header used by the Chrome Enterprise connector.
`xVerifiedAccessChallengeResponse`	A header used by the Chrome Enterprise connector.
`eventTimestamp`	The timestamp at which the most recent node in the flow was executed. For example, `1750864113753`.
`eventTimestampHumanReadable`	The timestamp at which the most recent node in the flow was executed in human-readable format. For example, `6/25/2025, 3:08:33 PM`.
`sessionToken`	The value of the session token cookie that PingOne uses to track the authentication session.
`sessionTokenMaxAge`	A value that controls the expiration of the PingOne session token cookie.
`p1UserId`	The PingOne User ID for the current user, if it is known.
`currentRequest > isSdk`	A boolean indicating whether the flow was launched with the SDK.
`currentRequest > platformType`	A string value indicating which platform invoked the SDK request.
`currentRequest > connection > ip`	The socket IP address of the connection used to submit the most recent request. Used when a Cloudflare custom domain has been configured in PingOne.

Parameter	Description
`currentRequest > connection > verified`	Indicates whether the connection used to submit the most recent request has been verified. This value is currently always set to false. This value is currently always set to `false`. Used when a Cloudflare custom domain has been configured in PingOne.
`currentRequest > connection > mtlsThumbprint`	The SHA-256 thumbprint, in lowercase hexadecimal, of the mTLS certificate used by the connection that submitted the most recent request. For standard TLS connections, this value is empty. Used when a Cloudflare custom domain has been configured in PingOne.
`currentRequest > connection > headers`	The custom headers submitted with the most recent request. Used when a Cloudflare custom domain has been configured in PingOne.
`language`	The user's language setting. This value comes from the user's browser language setting unless overridden by the Variables connector.
`parameters`	An object containing the parameters of the request made to invoke the flow, if the flow has an input schema.
`userInfo`	An object containing information about the user if the user has been identified.
`domainPublicHost`	A string containing the custom domain, if one is being used to access DaVinci.
`authentication`	An object containing the userId and authentication methods fulfilled by a PingOne capability, if any.

SK-Components reference

SK-Components are custom components that provide functionality and UI elements in the **HTML Template** field of connectors such as the HTTP connector.

Although these components are not required, they can provide unique functionality to your flows or speed up template building by providing preconfigured elements. Some common available elements provided by DaVinci's out-of-the-box SK-Components include buttons, error messages, input fields, and polling.

SK-Components look like rectangular pills in the **HTML Template** field, but get converted into raw HTML code when they are rendered in the widget.

You can also create your own custom HTML elements in the **HTML Template** using SK attributes. DaVinci recognizes these attributes to make static HTML function as SK-Components.

Adding SK-Components to a connector

Add an SK-Component to the **HTML Template** section of a connector such as the HTTP connector to add additional capabilities to the user-facing page for that node.

About this task

You can add SK-Components to multiple connectors, but this example adds an SK-Component to an HTTP node in your flow.

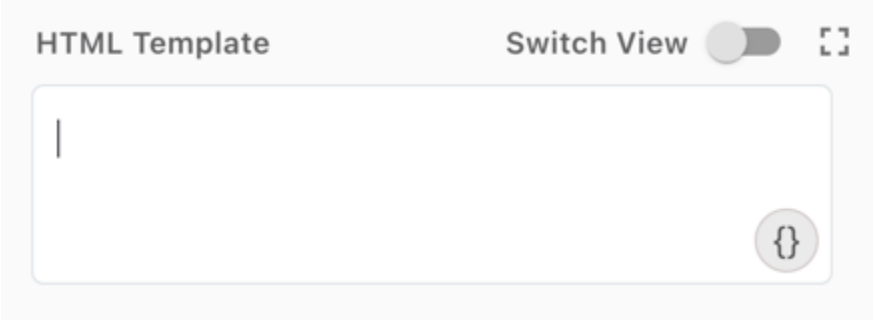
Steps

1. Add an HTTP connector to your flow.

   > **Note**
   >
   > For more information on adding a connector to your flow, see Adding a connector.

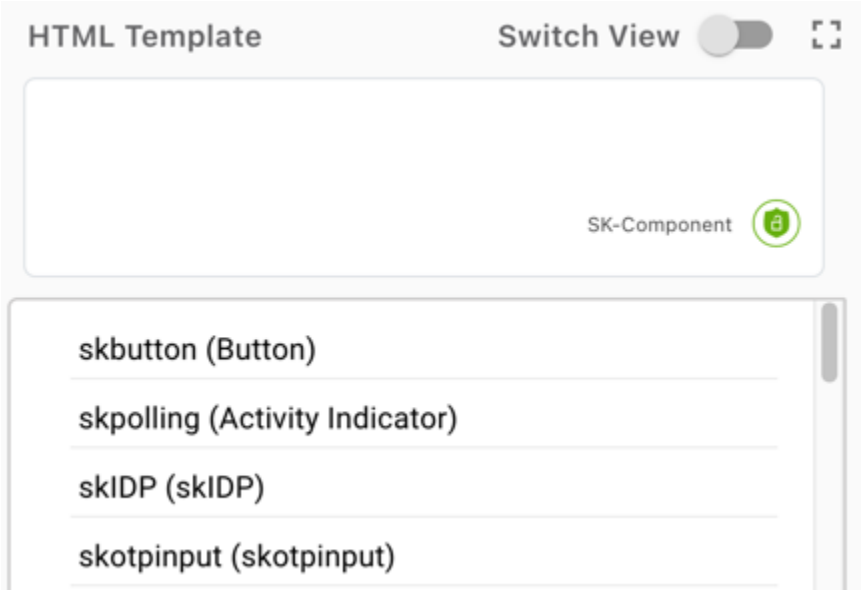
   A node is added to your flow.

2. In your flow, select the node that you added in the previous step.
3. Select the **Custom HTML Template** capability.
4. In the **HTML Template** section of the **General** tab, click the **{}** icon.



5. In the **Choose Connection** list, select **SK-Component**.

   You see a complete list of available components below the **HTML Template** section.



6. Select an SK-Component to include with your custom HTML for that page.

   For example, adding the `skbutton` component under your text displays a button for the user-facing page when the flow is run.

7. Click the component that appears in the **HTML Template** to view configuration options.

> **Note**
>
> SK-Components have different configuration options based on their functionality. For an overview of each component's configuration options, see SK-Components.

8. Click **Apply** after you finish adding your content and one or more SK-Components.

Next steps

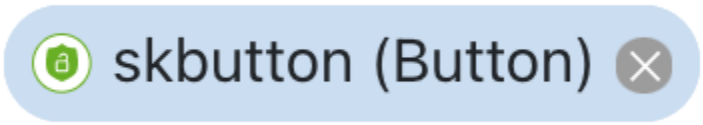
Click the **Save**, **Deploy**, and **Try Flow** buttons to view the user-facing page created by your custom HTML and test any SK-Components that you included.

SK-Components

Review the descriptions of each SK-Component and their configuration options.

> **Note**
>
> This document covers SK-Components from the UI. For more information on their raw HTML format, see SK attributes in custom HTML.

skbutton



The `skbutton` component is a generic button component that you can configure to perform various operations, such as form submit and next event.

At the most basic level, the button element will progress an end user to the next user-facing page created by the flow when clicked.

The following image shows what the component looks like on the user-facing page for the node.

The following table describes each configuration field on the **Identity Provider** tab.

Field	Description
Button ID	The ID of the button that serves as the main reference for any output the node can return.
CSS Class	The CSS class applied to the button element in the widget.

Field	Description
Label	The text that a user sees in a button on a user-facing page. If the field is left empty, the default label will be `Button`.
Button Value	The value of the button provides additional data that can be attached to a submit event that might not be defined in the form fields.

On the **Image** tab, there are additional configuration options, described in the following table.

Field	Description
Button Image (URL for an image)	Configures a custom button image with a URL.
Button Image Class	The CSS class applied to the button image that you can define in the **Custom CSS** section in the flow or by an external CSS file linked in the flow settings.
Button Image Placement	The position of the button on the user-facing page.

On the **Event** tab, there are additional configuration options, described in the following table.

Field	Description
Button Type	The function of the button. For example, `form-submit`.
Form ID	Field to enter the ID of the form tied to the button.
OTP Input ID	Field to enter the ID of the OTP input field being tied to the button.

On the **Loading** tab, there are additional configuration options, described in the following table.

Field	Description
Color of Default Loading Indicator	The color of the loading indicator when the button is pressed.
Custom Loading Indicator Image URL	A custom loading indicator configured through a valid URL for when a user clicks the button.
Custom Loading Indicator Image Class	The class of the custom loading indicator added through a valid URL when the button is pressed.

skpolling

The `skpolling` component displays a loading animation component that is tied to a polling function defined by a challenge in the flow details. The `skpolling` component continues polling until the challenge status changes.

The following image is what the component displays on the user-facing page for the node.

The following table describes each configuration field on the **Identity Provider** tab.

Field	Description
CSS Class	The CSS class applied to the indicator icon in the widget.
Poll Interval	The duration between polling functions that the widget will run in milliseconds. The default value is `2000`.
Poll Retries	The number of times the widget will poll the challenge. The default value is `60`.
Poll Challenge Status	Select **True** or **False** to turn polling on or off.

skIDP

The `skIDP` component is a **Login** button component you can use to trigger IdP logins as an alternative to using individual IdP connectors.

You can also use the `skIDP` component to reference external IdPs defined in PingOne in the **Integrations → External IDPs** section. For more information, see Identity providers. To reference an external IdP, add the component to a **Custom HTML Template** and select **PingOne Authentication** in the **Identity Provider Connector** field.

The following image is what the component displays on the user-facing page for the node.

The following table describes each configuration field on the **Identity Provider** tab.

Field	Description
Identity Provider Connector	Selects an identity provider. Select a connector in your flow to act as an IdP. You can also select **PingOne Authentication** to reference an external identity provider.
PingOne External Identity Provider	Select an external identity provider from your PingOne environment. **Note** This field is only shown when **PingOne Authentication** is selected to use an external identity provider.
PingOne External identity Provider ID	The ID for the external identity provider from your PingOne environment. **Note** This field is only shown when **PingOne Authentication** is selected to use an external identity provider.
Link with PingOne Use	When enabled, DaVinci creates or updates a linked PingOne user account using attributes from the external IdP. **Note** This field is only shown when **PingOne Authentication** is selected to use an external identity provider.
PingOne Population	The PingOne population to use when authenticating a user. **Note** This field is only shown when **PingOne Authentication** is selected to use an external identity provider.
Population ID	The ID of the PingOne population to use when authenticating a user. **Note** This field is only shown when **PingOne Authentication** is selected to use an external identity provider.
Button Text	The text that a user sees in the button on a user-facing page. If the field is left empty, the default label will be `Login`.
Button ID	The ID of the button that serves as the main reference for any output the node can return.

Field	Description
Button CSS Class	The CSS class applied to the button that can be referenced in other nodes in the same flow.

On the **Image** tab, there are additional configuration options, described in the following table.

Field	Description
Button image (URL for an Image)	Configures a custom button image with a URL.
Button Image Class	The CSS class applied to the button image that can be defined in the **Custom CSS** section in the flow or by an external CSS file linked in the flow settings.
Button Image Placement	The position of the button when displayed on the HTML page.

skthirdpartyiframe

The `skthirdpartyiframe` component is used with specific connectors, such as the Jumio connector, to render custom screens. For example, when used with the Jumio connector, the component renders a Jumio authentication screen within an iframe.

The following table describes the configuration fields on the **Identity Provider** tab.

Field	Description
Class	The CSS class applied to the iframe that can be defined in the **Custom CSS**section in the flow or by an external CSS file linked in the flow settings.
Width	The width of the iframe in pixels. The default width is `600px`.
Height	The height of the iframe in pixels. The default height is `600px`.

skerror

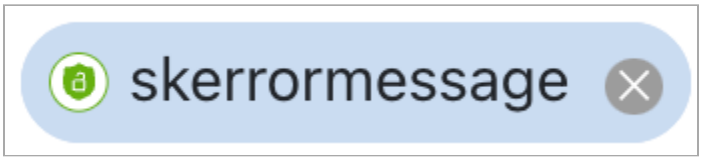
The `skerror` component is used to show validation error messages tied to specific input fields in a form.

By default, this element is not rendered. The error message is dynamically generated based on the validation message for that input field.

The following table describes the configuration fields on the **Identity Provider** tab.

Field	Description
Class	The CSS class applied to the error message that can be referenced in other nodes within the flow.
ID of input field associated with error message	The ID of the input field that you want to display an error message for under particular conditions.

skerrormessage



The `skerrormessage` component is used to show validation error messages tied to specific input fields in a form.

By default, this element is not rendered. The error message is dynamically generated based on the validation message for that input field.

The following table describes the configuration fields on the **Identity Provider** tab.

Field	Description
Class	The CSS class applied to the error message that can be referenced in other nodes within the flow.
ID of input field associated with error message	The ID of the input field that you want to display an error message for under particular conditions.

skinputfile

The `skinputfile` component is a **Choose File** button component that allows end users to upload files to a page.

The following image is what the component displays on the user-facing page for the node.

The following table describes the configuration fields on the **Identity Provider** tab.

Field	Description
Class	The CSS class applied to the input field element.
ID of component that will be associated with the base64 value of the file	The ID of the component that will be associated with the base64 value of the file.

On the **Preview** tab, there are additional configuration options, described in the following table.

Field	Description
Show Preview	Configures whether or not a preview is displayed when adding files.
Image	Configures between `Text` and `Image` options.
Class of span added next to input button when file is set	The CSS class applied to the span added next to the input button when a file is uploaded.
Class of image added next to input button when file is set	The CSS class applied to the image added next to the input button when a file is uploaded.
Width of preview image in pixels	The width of the preview image in pixels. The default width is `100` pixels.
Height of preview image in pixels	The height of the preview image in pixels.

skrecaptcha

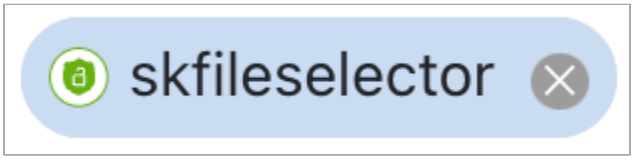
The `skrecaptcha` component is a reCAPTCHA component that validates the end user is not a bot.

The following table describes the configuration fields on the **Identity Provider** tab.

Field	Description
Class	The CSS class applied to the reCAPTCHA element.
ID of component that will be associated with the recaptcha value	The ID of the component that will be associated with the reCAPTCHA value and that can be referenced in other nodes within the flow. The autofilled ID is `recaptchaValue` and cannot be changed.

Field	Description
Recaptcha Site Key	The site key for the reCAPTCHA element to include in the deployed flow. **Note** A site key must be included in this field for the reCAPTCHA element to display. For more information on creating a site key to use in this field, see https://cloud.google.com/recaptcha-enterprise/docs/create-key.

skfileselector



The `skfileselector` component is a PDF and image file selector component intended to be used with a form submit view.

The following image is what the component displays on the user-facing page for the node.

The user interface for the file selector includes a **Select File** and **Open Camera** button. Additionally, there are **Select All** and **Unselect All** buttons for the end user to manage the files they have added before submitting.

After a user uploads files, a **Remove** button appears on each file.

Clicking the component followed by the **General** tab displays the **PDFJS CDN Url** field, which is the only configuration option for this component. This field loads a PDFJS library used to process selected PDFs and extract individual pages from it.

On the **View Classes** tab, there are additional configuration options, described in the following table.

Field	Description
Root View class	The CSS class applied to the root view element on the file selector.
Input file button class (hidden by default)	The CSS class applied to the input file button, which is hidden by default.
Input file container class	The CSS class applied to the input file container element.

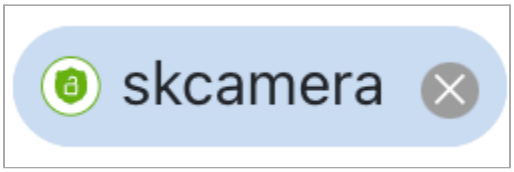
Field	Description
Input field label (Shown instead of input field by default)	The CSS class applied to the display name that appears for the input field.
Main file view class	The CSS class applied to the main file view element.
File list view class	The CSS class applied to the file list view element.
File preview class	The CSS class applied the file preview element.
File name shown inside file preview	The CSS class applied to the file name shown inside the file preview.
File preview image	The CSS class applied the file preview image.
Button view class containing buttons	The CSS class applied to the button view element containing the **Remove**, **Select All**, **Unselect All**, and **Submit** buttons.
Remove button class inside file preview	The CSS class applied to the **Remove** button.
Select All button in bottom left	The CSS class applied to the **Select All** button.
Unselect button in bottom right (Hidden by default)	The CSS class applied to the **Unselect All** button.
Submit button in bottom right (Hidden by default)	The CSS class applied to the **Submit** button.
Bottom left button container	The CSS class applied to the container for the **Select All** button.
Bottom right button container	The CSS class applied to the container for the **Unselect All** button.

On the **Messages** tab, there are additional configuration options, described in the following table.

Field	Description
Input label content	The text that users see in the input field element on the user-facing page. If the field is left empty, the default label is `Select File`.
Remove button label	The text that users see in the **Remove** button element on the user-facing page. If the field is left empty, the default label is `Remove`.
Select All button label	The text that users see in the **Select All** button element on the user-facing page. If the field is left empty, the default label is `Select All`.
Unselect All button label	The text that users see in the **Unselect All** button element on the user-facing page. If the field is left empty, the default label is `Unselect All`.

Field	Description
Submit button label	The text that users see in the **Submit** button element on the user-facing page. If the field is left empty, the default label is `Submit`.

skcamera



When used with the TransUnion TLOxp connector, the `skcamera` component embeds a camera element for an end user to take pictures from their device and upload them.

The TransUnion TLOxp connector lets you verify a user's identity information in your flow by checking TransUnion's trusted data sources. For more information, see the TransUnion TLOxp connector documentation.

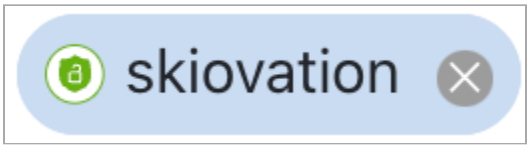
The following table describes the configuration fields on the **Identity Provider** tab.

Field	Description
Capture Message	The message displayed when a picture is captured by the camera element in a user-facing page.
Capture Message	The message displayed when a picture is captured by the camera element in a user-facing page.

On the **View Classes** tab, there are additional configuration options, described in the following table.

Field	Description
Root Class	The CSS class applied to the root element for the camera.
Video Class	The CSS class applied to the video element for the camera.
Button Container	The CSS class applied to the **Capture** button container for the camera.
Capture Button	The CSS class applied to the **Capture** button element for the camera.
Camera Canvas	The CSS class applied to the camera canvas element for the camera.

skiovation



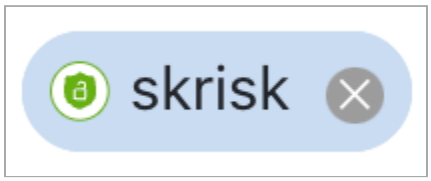
The `skiovation` component is an Iovation script loader to get a blackbox value, an encrypted string generated by the Iovation connector.

An Iovation connector added to your flow performs device risk checks and OTP during enrollment, authentication, and other transactions.

The following table describes the configuration fields on the **Identity Provider** tab.

Field	Description
Iovation Loader CDN Link	The CDN URL for the Iovation script loader element.
Iovation Loader Version	The loader version for the Iovation script loader element.
Iovation Loader Sub Key	The sub key for the Iovation script loader element.
Blackbox Property Name	The name of the blackbox value of the Iovation script loader element that can be referenced in other nodes of the same flow.

skrisk



The `skrisk` component can be added to HTML forms to provide access to browser data for risk signals used in the PingOne Protect connector.

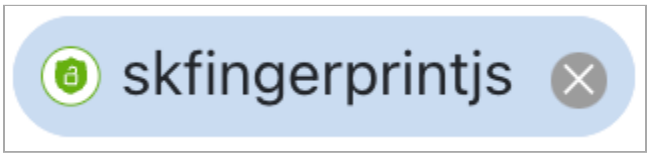
Including the `skrisk` component in an HTTP node placed before the relevant PingOne Protect connector automatically obtains additional risk-related variables without having to write the usual code in your **HTML Template** field.

The following table describes the configuration fields on the **Identity Provider** tab.

Field	Description
Environment ID	The environment ID of your PingOne environment.
Risk Property Name	The name assigned to the Risk property.

> **Note**
>
> See the PingOne Protect connector documentation for instructions on adding the PingOne Protect connector and `skrisk` component to your flow.

skfingerprintjs

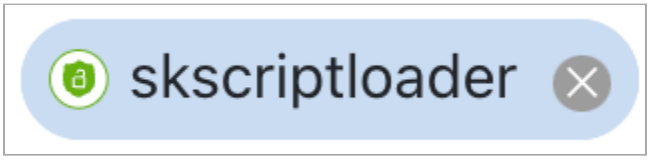


The `skfingerprintJS` component is a script to get a browser fingerprint value. Add it to HTML forms to identify end user behavior with the FingerprintJS connector.

The following table describes the configuration fields on the **Identity Provider** tab.

Field	Description
FingerprintJS CDN Link	The CDN URL for the FingerprintJS element.
Fingerprint JS Browser Token	The browser token for the FingerprintJS element.
FingerprintJS Property Name	The name assigned to the FingerprintJS property that can be referenced in other nodes in the same flow.

skscriptloader



The `skscriptloader` component is an SK script loader to load any external scripts.

Clicking the component displays the **CDN Link** field on the **Identity Provider** tab, which is the only configuration option for this component. This field is for configuring the CDN URL for the script loader element.

SK attributes in custom HTML

SK attributes are data attributes that DaVinci recognizes and acts upon. Adding SK attributes to the static HTML elements in your **Custom HTML** section makes the components interactive components in a flow.

> **Note**
>
> This document covers SK-Components in raw HTML format and their SK attributes. For more information on how they function as part of the UI, see SK-Components.

You can create custom components in the **Custom HTML** field. Use a `<div>` or `<form>` wrapper for the component, then add data attributes to provide functionality. The first data attribute in custom HTML should be the `data-skcomponent` attribute, which invokes the relevant SK-Component. For example, `data-skcomponent="skbutton"` invokes the `skbutton` component.

> **Important**
>
> If you are writing or importing the custom HTML for SK-components, data attributes with meaningful classes and IDs are important because they serve as the main reference points for managing node outputs within your flow.

skbutton HTML

A complete raw HTML for the `skbutton` component looks like this:

```
<button
  type="button"
  data-skcomponent="skbutton"
  id="<some_id>"
  class="<some_class>"
  data-skvalue="value-sent-with-button-submit"
  data-skbuttonvalue="value-sent-with-button-submit"
  data-skbuttontype="form-submit|mfa-reset|otp|next-event|nuance-audio|change-view"
  data-skviewindex="1"
  data-skform="form-id-to-submit"
  data-skotpinput="otp-input-field-id-to-submit"
  data-skbuttonimage="image-src|data-url-to-show-inside-button"
  data-skbuttonimageplacement="left|right"
  data-skbuttonimageclass="button-image-class"
  data-skdefaultloadingcolor="loading-indicator-color"
  data-skcustomloadingindicator="loading-indicator-url"
  data-skcustomloadingindicatorclass="loading-indicator-class"
  >
  Button Label
</button>
```

This also works with an `<input>` tag with the `type="submit"` attribute included:

```
<input
  type="submit"
  data-skcomponent="skbutton"
  id="<some_id>"
  class="<some_class>"
  data-skbuttonvalue="value-sent-with-button-submit"
  data-skbuttontype="form-submit|mfa-reset|otp|next-event|nuance-audio|change-view"
  data-skviewindex="1"
  data-skform="form-id-to-submit"
  data-skotpinput="otp-input-field-id-to-submit"
  data-skbuttonimage="image-src|data-url-to-show-inside-button"
  data-skbuttonimageplacement="left|right"
  data-skbuttonimageclass="button-image-class"
  data-skdefaultloadingcolor="loading-indicator-color"
  data-skcustomloadingindicator="loading-indicator-url"
  data-skcustomloadingindicatorclass="loading-indicator-class"
  value="Button Label"
  >
</input>
```

The following table provides descriptions for data attributes that can be used with the raw HTML for the button component.

Attribute	Description
`data-skbuttonvalue`	The value added to the form submit payload.
`data-skbuttontype`	The event assigned to the button.
`data-skform`	The ID of the form to be submitted using `form-submit` button type.
`data-skotpinput`	The ID of the input field used for OTP value.
`data-skbuttonimageplacement`	Placement of the image with respect to the label. Valid values are `left` or `right`.
`data-skbuttonimageclass`	The class name assigned to the `<img>` tag for the `data-skbuttonimage` attribute. This attribute styles the image.
`data-skdefaultloadingcolor`	The color of the loading spinner created by the `data-skcustomloadingindicator` attribute.
`data-skcustomloadingindicator`	The URL of a spinner image that is used instead of the default loading icon when the button is submitted.
`data-skcustomloadingindicatorclass`	The class name assigned to the `<img>` tag for the `data-skcustomloadingindicator` attribute. This class name styles the image.

The following table provides descriptions for the various types of buttons.

Type	Description
`form-submit`	Submits a form. Used alongside `data-skform` whose value is used as the ID for getting form values.
`mfa-reset`	Resets MFA dialogue box back to **selection** page.
`otp`	Submits OTP input field value. The OTP input value will be assigned to a property named `OTP`.
`next-event`	Continues to the next screen without submitting input values. The `data-skbuttonvalue` attribute will progress to the next page.
`nuance-audio`	Controls Nuance audio mic recording.
`change-view`	Switches between views in a multiview screen. Used with the `data-skviewindex` attribute, whose value will be the view index that the widget will switch to.

skpolling HTML

The raw HTML for the `skpolling` component looks like this:

```
<div data-skcomponent="skpolling"
  class="<some_class>"
  data-skpollinterval="2000"
  data-skpollretries="60">
</div>
```

skIDP HTML

The raw HTML for the `skIDP` component looks like this:

```
<button data-skcomponent="skIDP"
  class="<some_class>"
  data-skidpvalue="<some_idp>"
  data-skbuttonimage="<some_image>"
  data-skbuttonimageplacement="<some_placement>"
  data-skbuttonimageclass="<some_class>">Login text
</button>
```

skthirdpartyiframe HTML

The raw HTML for the `skthirdpartyiframe` component looks like this:

```
<div data-skcomponent="skthirdpartyiframe"
  class="<some_class>"
  data-skwidth="640"
  data-skheight="480">
</div>
```

skerrormessage HTML

The raw HTML for the `skerrormessage` component looks like this:

```
<div data-skcomponent="skerrormessage"
  class="<some_class>"
  data-skerrorid="<some_id>">
</div>
```

The `data-skerrorid` attribute is the ID of the input field that the error message is associated with.

skinputfile HTML

The raw HTML for the `skinputfile` component looks like this:

```
<input data-skcomponent="skinputfile"
  type="file"
  class="<some_class>"
  data-skinputid="<some_id>"
  data-skshowpreview="yes"
  data-skpreviewtype="text"
  data-skpreviewtextclass="<some_class>"
  data-skpreviewimageclass="<some_class>"
  data-skpreviewimagewidth="640"
  data-skpreviewimageheight="480"
  data-skpdfjsurl="<pdf_js_cdn_url>">
</input>
```

skrecaptcha HTML

The raw HTML for the `skrecaptcha` component looks like this:

```
<div data-skcomponent="skrecaptcha"
  class="<some_class>"
  data-skcaptchaid="<some_id>"
  data-skrecaptchasitekey="<some_site_key>">
</div>
```

skfileselector HTML

The raw HTML for the `skfileselector` component looks like this:

```
<div data-skcomponent="skfileselector"
  data-sk_<some_prop_to_pass>="<your_value>">
</div>
```

skcamera HTML

The raw HTML for the `skcamera` component looks like this:

```
<div data-skcomponent="skcamera"
  data-skroot="sk-camera-root"
  data-skvideo="liveView"
  data-skbtnactions="sk-camera-btn-actions"
  data-skbtncapture="sk-camera-btn-capture"
  data-skcanvas="liveView"
  data-skcapturemessage="Capture Image"
  data-skswitchcameramessage="Switch Camera Message">
</div>
```

skiovation HTML

The raw HTML for the `skiovation` component looks like this:

```
<div data-skcomponent="skiovation"
  data-sk_<some_prop_to_pass>="<your_value>">
</div>
```

skrisk HTML

The raw HTML for the `skrisk` component looks like this:

```
<div data-skcomponent="skrisk"
  data-sk_<some_prop_to_pass>="<your_value>">
</div>
```

skfingerprintjs HTML

The raw HTML for the `skfingerprintjs` component looks like this:

```
<div data-skcomponent="skfingerprintjs"
  data-sk_<some_prop_to_pass>="<your_value>">
</div>
```

skscriptloader HTML

The raw HTML for the `skscriptloader` component looks like this:

```
<div data-skcomponent="skscriptloader"
  data-sk_<some_prop_to_pass>="<your_value>">
</div>
```

Viewing flow analytics

View detailed information about a flow's run history.

Steps

1. Click the **Flows** tab.
2. Find the flow and click **... → Edit**.
3. Click **Analytics**.
4. **Optional:** On the **Analytics** tab, select a date range from the list.

   The graph displays the flow runs within the date range, and the **Events Logs** section shows the logs for individual runs.
5. **Optional:** Search for specific flow executions.
   1. Select a search parameter.
      - **Flow Execution ID**
      - **User ID**
      - **User Name**
      - **Correlation ID**
   2. Enter a search term corresponding to the selected search parameter.
   3. Press `Enter`.
6. If the list of executions contains more than one result, select a **Flow Execution ID** to view details of the selected run.

   The nodes that ran during the selected run are highlighted. The **Events Logs** section shows the events that occurred during the run.

   To see logs related to execution of a specific node, click on that node. This highlights the associated entry in the events log pane.

   > **Note**
   >
   > The node's records often include two entries for the node's execution, corresponding to the payload sent to the node and the response from the node.
7. Select an event to view additional event details.
8. Click **X** to close the **Analytics** tab.

Using load balancing

Use load balancing to run different branches in a flow on different executions.

Steps

1. Click the **Flows** tab.
2. Locate and open the flow.
3. Click the **Action Decision** node.
4. Select **Load Balance** to enable load balancing.
5. In the **Load Balance Context** list, select a context for load balancing.

   The context determines how widely the algorithm is applied. For example, a Round Robin algorithm with a local context will always begin with the first option and only use the second option if the flow returns to the same node again. However, a Round Robin algorithm with the Global context will consider which option was used in the previous flow execution.

   Choose from:

   - **Global**: The algorithm is applied using data from all flow executions.
   - **Local**: The algorithm is applied using only data from the current flow execution.
6. In the **Load Balance Algorithm** list, select an algorithm to use for load balancing.

   Choose from:

   - **Random**: Uses a randomly selected exit path.
   - **Random Weighted**: Uses a randomly selected exit path, with the probability of each path determined by its weight.
   - **Round Robin**: Cycles through the available exit paths, with each execution using the option after that of the previous execution.
7. For each path out of the node, select a logical operator to determine when it should be considered by the load balance algorithm.

   Choose from:

   - **All Triggers True**: All of the nodes leading into the action decision node completed successfully.
   - **All Triggers False**: All of the nodes leading into the action decision node failed.
   - **Any Triggers True**: One or more of the nodes leading into the action decision node completed successfully.
   - **Any Triggers False**: One or more of the nodes leading into the action decision node failed.
   - **Any Trigger Completes**: One or more of the nodes leading into the action decision node completed either successfully or unsuccessfully.
   - **All Triggers Complete**: All of the nodes leading into the action decision node completed either successfully or unsuccessfully.
8. If you selected the **Random Weighted** algorithm, select a weight for each path out of the node.

9. Click **Save**.

Using flow search

Use the search option to find specific nodes within the current flow.

When you open a flow, the search option displays in the upper left. You can enter a search term to locate nodes using any of the following parameters:

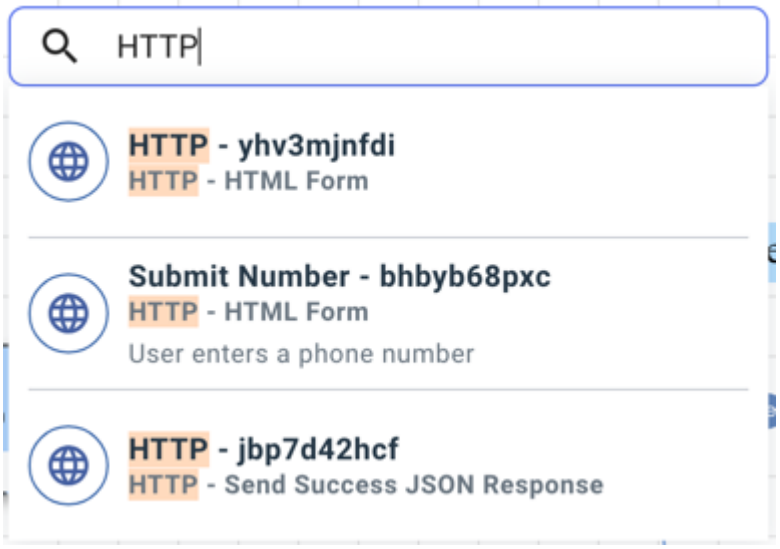
- **Connector name**: A connector's official name.
- **Connector instance name**: The name used in your environment for a specific connector configuration.
- **Capability**: A specific connector capability.
- **Node title**: The title of a node.
- **Node description**: The description value of a node.
- **Node ID**: The unique ID assigned to a node.

Steps

1. In DaVinci, on the **Flows** tab, locate and open the flow.
2. Click the search icon and enter a search term.

   The nodes that match the search term appear in a list.

3. Click a node from the list.

   The flow canvas shifts to the selected node and displays the node details.

Editing the input schema

Configure the information that's provided to the flow when it is invoked. This lets you map in external parameters for dynamic content or logic.

Steps

1. Click the **Flows** tab.
2. Find the flow and click **... → Edit**.
3. Click **Input Schema**.
4. On the **Input Schema** tab, add one or more properties to the input schema:
    1. Click **Add** to add a new property.
    2. In the **Parameter Name** field, enter a name for the input parameter.

        > **Note**
        >
        > The name `challenge` is reserved and cannot be used.

    3. **Optional:** In the **Description** field, enter a description for the input property.
    4. In the **Data Type** list, select a data type for the input property.
    5. Select **Required** if the property is required for the flow.
5. **Optional:** Click **All Required?** to change the **Required** setting of all properties.
6. **Optional:** Click **Edit** to reorder or remove properties, and then click **Close**.

Option	Description
Delete	Click to remove a property.
Reorder	Click and drag to reorder a property.

7. Click **Save**.

Editing the output schema

Configure the information included in the output when the flow completes.

About this task

The output from the flow is generated using a JSON schema with `output` as a root object.

Steps

1. Click the **Flows** tab.

2. Find the flow and click **...** > **Edit**.
3. Click **More options (⋮)** > **Output Schema** to show the output schema.
4. Update the JSON schema.

   You can add one or more parameters to the properties section. For example, this output schema outputs the errorMessage and selectedDeviceOtpEnabled properties:

```
{
  "output": {
    "type": "object",
    "additionalProperties":true,
    "properties": {
      "errorMessage": {
        "type": "string",
        "displayName": "Error Message",
        "preferredControlType": "textField",
        "enableParameters": true,
        "propertyName": "errorMessage"
      },
      "selectedDeviceOtpEnabled": {
        "type":"bool"
      }
    }
  }
}
```

5. Click **Save**.

Editing flow settings

Configure the settings that apply to the flow as a whole.

Steps

1. Click the **Flows** tab.
2. Find the flow and click **...** > **Edit**.
3. Click **More options (⋮)** > **Flow Settings** to show the flow settings.
4. (Optional) Click the **General** tab to configure general settings for the flow.

Option	Description
PingOne Flow	Indicates that the flow is a PingOne flow, enabling it to be included in PingOne flow policies and launched directly from PingOne.
Validate Flows when Changes are Saved	Automatically validate the flow whenever it is saved.
Require Authentication to Initiate Flow	Require authentication as part of the flow initiation.

Option	Description
Flow Timeout (in seconds)	A timeout value for the entire flow, beginning when the flow is invoked. The default value is 300 (five minutes) and the maximum value is 172800 (two days).
Flow HTTP Response Timeout (in seconds)	A timeout value for any HTTP call made by any node in the flow. The default value is 15 and the maximum value is 120.
Enable Content Security Policy	Restricts the domains from which content can be loaded.
CSP Value	If you selected **Enable Content Security Policy**, enter your content security policy in text form.

5. (Optional) Click the **Logging** tab to configure logging levels and sensitive data protections for the flow.

Option	Description
Log Level	The logging level for the flow. The options are: ◦ **None**: Prevents logging ◦ **Info**: Logs info-level and more severe events ◦ **Debug**: Logs debug-level and more severe events
Scrub Sensitive Information	Remove fields designated as sensitive from analytics.
Sensitive Information Fields	If you selected **Scrub Sensitive Information**, enter one or more fields to designate as sensitive.

6. **Optional:** Click the **Customizations** tab to configure UI customizations for the flow.

Option	Description
Page Title	An HTML page title to use in place of the default title.
Favicon URL	A link to a favicon for the flow.
Error Page Logo	A list for selecting the logo to display on the error page. The options are: ◦ **None**: Displays no logo. ◦ **Ping Identity Logo**: Displays a Ping Identity logo. ◦ **Use Custom Logo URL**: Displays a logo from a URL that you specify using the **Custom Logo URL** field.
Custom Logo URL	The URL for a brand logo to use on error screens if **Use Custom Logo URL** is selected in the **Error Page Logo** list.

Option	Description
Show Footer On Error Screen	Show the DaVinci footer on error screens.
Use Custom CSS	Apply custom CSS to the user-visible nodes in the flow.
Custom CSS Rules	If you selected **Use Custom CSS**, enter the CSS code to apply.
Custom CSS files	If you selected **Use Custom CSS**, enter one or more links to the CSS code to apply.
Use Custom JavaScript	Apply a custom JavaScript to the user-visible nodes in the flow.
Custom JavaScript files	If you selected **Use Custom Script**, enter one or more links to JavaScript files to apply to the user-visible nodes.
Show Intermediate Loading Page	Display an intermediate screen between nodes in the flow.
Intermediate Loading Page HTML	Enter HTML for the loading screen.
Intermediate Loading Page CSS Rules	Enter CSS for the loading screen.

7. Click **Save**.

Adding annotations

Add annotations to make a flow more comprehensible to yourself and others developing the flow.

Steps

1. Click the **Flows** tab.
2. Find the flow and click **... → Edit**.
3. Right-click the flow canvas.
4. Click **Add Annotation**.
5. Click the annotation.
6. **Optional:** Update the annotation properties.
    1. In the **Annotation Text** field, enter the text to display in the annotation.
    2. In the **Background Color** selector, select a color for the annotation background.
    3. In the **Text Color** selector, select a color for the annotation text.
    4. In the **Width** field, enter a width in pixels for the annotation.

5. In the **Height** field, enter a height in pixels for the annotation.
6. Select **Stroke Enabled** to add a border around the annotation.
7. In the **Stroke Width** field, enter a width in pixels for the stroke.
8. In the **Stroke Color** selector, select a color for the stroke.
9. In the **Corner Radius** field, enter a radius in pixels for the rounding of the annotation corners.
10. In the **Font Size** field, enter a font size for the annotation text.
11. In the **Font Style** selector, select a font style. Valid options are **Normal**, **Bold**, and **Italic**.
12. In the **Font Family** field, enter a font for the annotation text.

7. Click **Apply**.

Validating a flow

Validate a flow to automatically search for problems within the flow. These problems fall into one of two categories:

- **Error**: A problem that will prevent the flow from completing successfully and must be fixed before you deploy the flow.
- **Warning**: A problem that won't prevent the flow from completing successfully but could make it difficult to understand or update the flow.

When you validate a flow, the errors and warnings are recorded with the current flow version. If you reopen a flow version that you have already validated, you can click **View X Error(s)** or **View X Warning(s)** to view the errors and warnings.

The list of flows displays an icon indicating the number of errors in the current flow version, or the number of warnings if the flow contains no errors.

> **Note**
>
> You can configure a flow to be automatically validated whenever it is saved. Learn more in Editing flow settings.

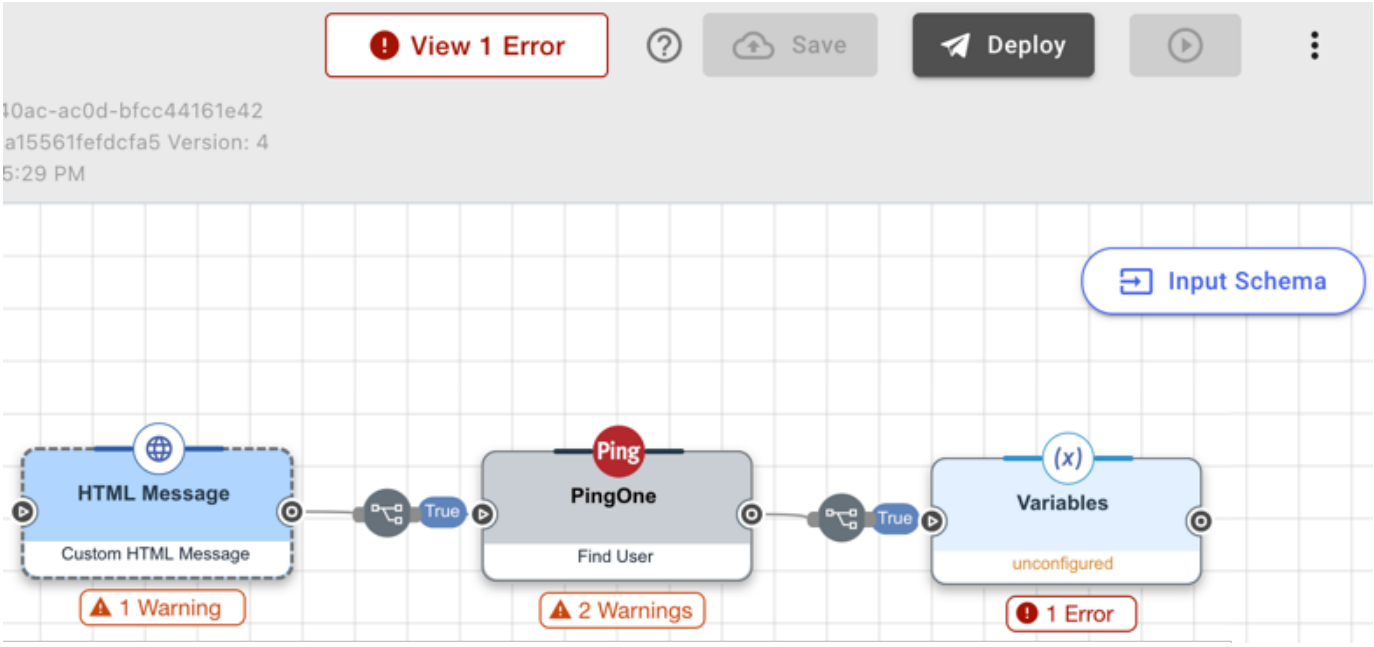
To validate a flow manually:

Steps

1. In DaVinci, on the **Flows** tab, locate and open the flow.
2. Click **Validate Flow**.

   If the flow has errors or warnings, the top of the center pane displays one of the following:

   - **View X Error(s)**: Displays if the flow has one or more errors.
   - **View X Warning(s)**: Displays if the flow has no errors but one or more warnings.



3. Click **View X Error(s)** or **View X Warning(s)**.

   The **Error Validation** pane displays.

4. Select **Show Warnings** to include warnings in the display.

5. For each error or warning, read the description to understand how to fix the issue. Click **View Error** or **View Warning** to highlight the node containing the issue.

6. Click **Save** after you have fixed the issues.

Flow Validation Rules

Errors

The following errors can be identified by flow validation:

Error	Description
Flow is empty	The flow does not contain any nodes.
Flow has multiple start points or a floating node	There are multiple nodes that could act as starting points for the flow.
Disabled node found	The flow contains one or more disabled nodes, which can cause issues when a flow is run.
Subflow configuration error	The flow launches a subflow, but the subflow or subflow version are not valid.
Circular subflow dependency found	The flow launches a subflow, but that subflow launches the parent flow, creating an infinite loop.

Error	Description
Unused variable found	A flow instance variable is defined by a variable connector but is not used in the flow.
Undefined variable found	A variable that is not defined in the flow is being referenced.
Subflow input schema missing	The flow launches a subflow, but the flow conductor node does not provide one or more values that are required by the subflow input schema.
Incorrect ending nodes for PingOne flow	The flow is a PingOne flow, but it includes branches that do not conclude with a PingOne Authentication connector node using either the **Return Success Response (Redirect Flows)** capability or the **Return Error Response (Redirect Flows)** capability.
Form not selected	The flow includes a **Forms** connector node that doesn't have a form selected.
Connector capability not configured	The flow contains a node that does not have a selected capability.
Referenced node in local variable doesn't exist	The flow contains a node that references a value from an unavailable node ID.

Warnings

The following warnings can be identified by flow validation:

Error	Description
Log level set to Debug	The log level for the flow is set to **Debug**. You should only use this log level if you are actively troubleshooting an issue. Learn more in Editing flow settings.
Missing node title	A node has no title, which makes identifying the node challenging.
Missing node description	A node has no description, which makes identifying the node's purpose challenging.
Incorrect node color	A **Send Success Response** or **Send Error Response** node does not have the recommended node color. Using a consistent color for these nodes makes it easier to identify the flow's endpoints.

Renaming a flow

Rename a flow to give it a new name and description.

Steps

1. Click the **Flows** tab.

2. Find the flow you want to rename and click **... → Rename**.

   A confirmation modal displays.

3. In the **Name** field, enter a new name for the flow.
4. **Optional:** In the **Description** field, update the description of the flow.
5. Click **Rename**.

Cloning a flow

Clone a flow to create an identical new flow.

About this task

Cloning an existing flow lets you use the existing flow's steps and settings as a starting point for a modified flow.

Steps

1. Click the **Flows** tab.
2. Find the flow you want to clone and click **... → Clone**.

Result

A new flow is created with identical naming, steps, and settings. The description of the new flow indicates the time at which it was cloned.

Exporting a flow

Export a flow to create a flow backup file that you can analyze or import into another environment.

About this task

Exporting a flow creates a JSON version of the flow. You can use this file to:

- Copy flows between your different environments, such as between test and production
- Create backups or store flows in external version control systems
- Share the flow with colleagues who use a different environment

Ping support might also request flows for troubleshooting purposes.

Steps

1. In DaVinci, click the **Flows** tab.
2. Open the flow that you want to export.
3. Choose one of the export options:

   Choose from:

   - To export the current version, click **More options (⋮) > Download Flow JSON**.

- To export a previous version, go to **More options (⋮) > Flow Versions**, locate the version that you want to export, and click **... > Download Flow JSON**.

4. (Optional) Select **Include Variable Values** to include the current values of company or flow variables that are used in the flow.

> **Important**
>
> Because variables can contain secret or proprietary information, only use this option for flows that you do not plan to share outside of your organization.

5. (Optional) If the flow uses subflows, select **Include Subflows** to include the subflows in the JSON file.

   If you plan to import the flow into another environment for a second time, you can disable this option to avoid reimporting the subflows.

6. Click **Export**.

> **Note**
>
> If you selected **Include Subflows**, and the flow contains **Flow Conductor** nodes that use invalid subflows or subflow versions, an error message displays detailing the issues. You must resolve these issues before exporting the flow with its subflows.

Deleting a flow

Delete an existing flow.

About this task

> **Important**
>
> Verify that the flow you intend to delete is not in use.

Steps

1. Click the **Flows** tab.
2. Find the flow you want to delete and click **... → Delete**. On the confirmation modal, click **Delete**.

Flow limits

These limits affect the creation of flows.

Some connectors have additional limitations; see the connector documentation for your connectors to learn more.

For PingOne standard platform limits, see PingOne standard platform limits.

Flow limits

Property	Limit	Description
Resource	100	Your environment can contain a maximum of 100 of each entity type. This limitation applies to flows, variables, and connections, but not to end users. Each flow is limited to 100 stored versions.
API payload	10 MB	The maximum payload size for any API call (such as POST, GET, or DELETE) made within a flow.
Number of nodes that can be run per flow invocation	Twice the number of nodes in the flow or 70, whichever is higher	Maximum number of nodes that can be run. This limit is distinct for each flow, so if a flow launches a subflow, both flows can run a number of nodes up to this limit.
Flow timeout	Default is 5 minutes. Maximum possible is 2 days.	A timeout value for the entire flow, beginning when the flow is invoked. Editable for each flow. For more information, see Editing flow settings.
Flow HTTP response timeout	Default is 15 seconds. Maximum possible is 120 seconds.	A timeout value for any HTTP call made by any node in the flow. Editable for each flow. For more information, see Editing flow settings.
Node payload	1 MB	The maximum size of the payload sent by a node to future nodes.

DaVinci Best Practices

This document describes best practices for creating and maintaining flows in DaVinci.

DaVinci flows are a powerful tool for orchestrating your organization's identity services. However, because DaVinci flows are versatile, it can be difficult to create flows that work optimally for the user, for the people maintaining the flows, and from the perspective of security.

These best practices can help you avoid common problems, improve performance and reliability, and simplify future improvements.

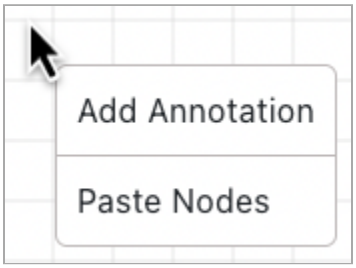
Building flows

Use these best practices when creating a flow.

Annotate flows

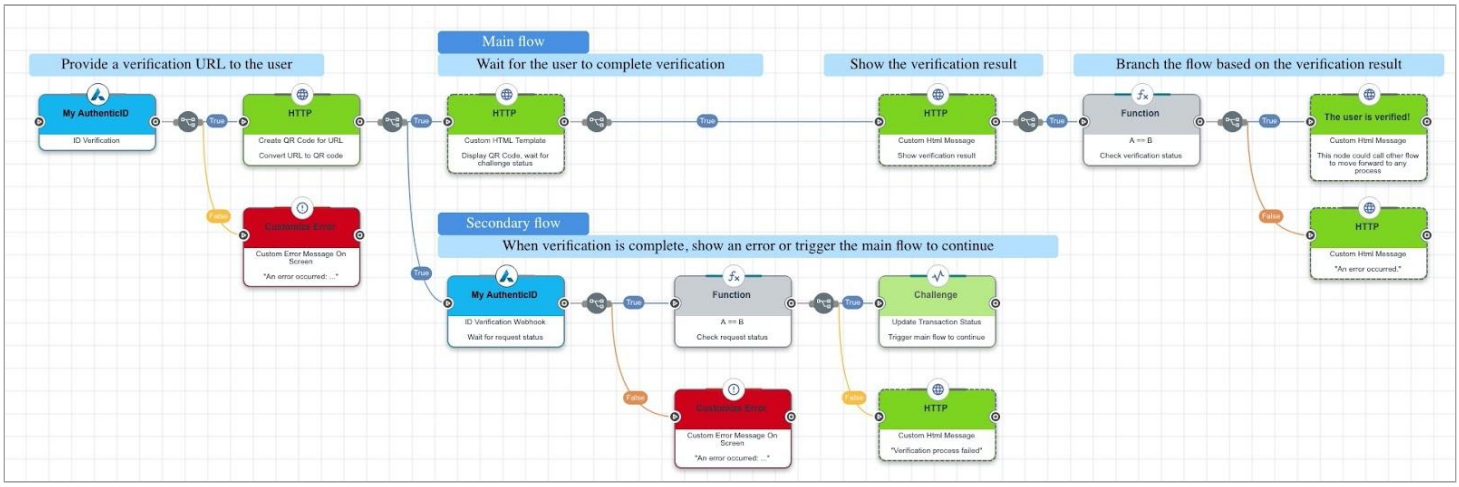
Annotate your flows to make them easy to understand at a glance. These annotations might benefit another flow builder in your organization, or they might benefit you if you haven't worked on the flow in a while.

To add an annotation to the flow canvas:



1. CMD+Click (macOS) or CTRL+Click (Windows) a blank part of the canvas.
2. Click **Add Annotation**.
3. Click the annotation to enter the text and configure the appearance.

Use annotations to describe what happens in the main steps of the flow. Although there are rare occasions where a single node should have its own annotation, in most cases, you should use one annotation for each logical grouping of nodes.

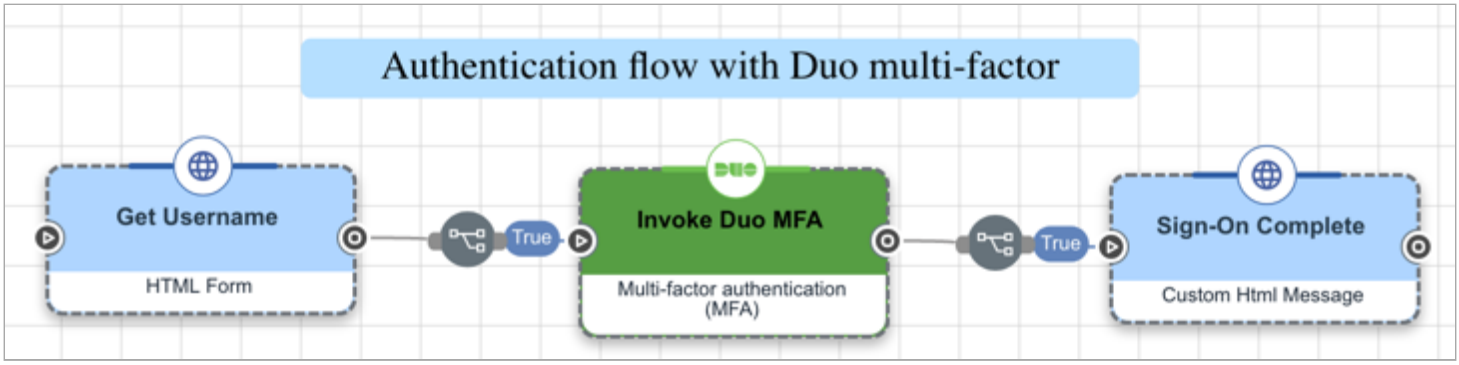


Add meaningful node titles

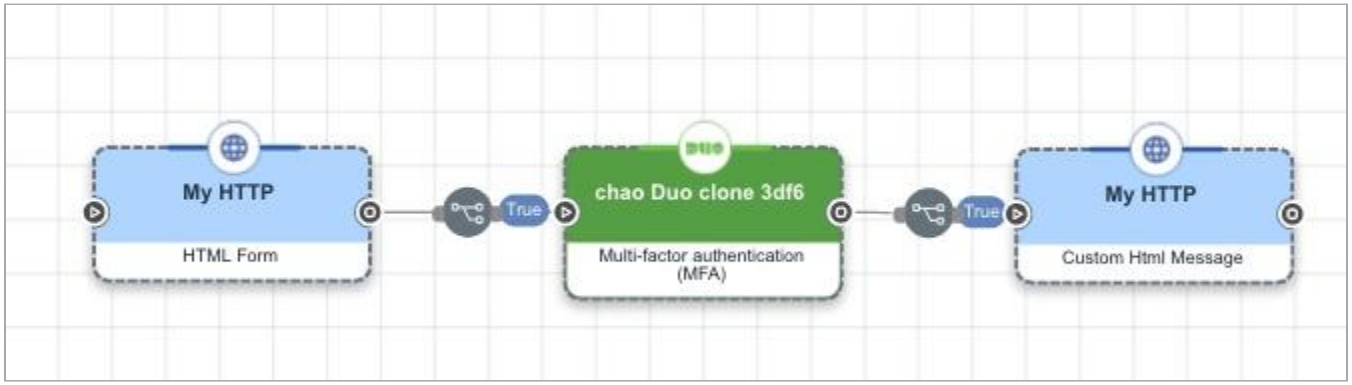
On the **Settings** tab for each node, set the **Node Title** to a meaningful name that describes or identifies each node. This name appears in the flow canvas and, combined with the annotations, makes it easier to read the flow.

For example, consider the following flows. They are identical aside from node titles and annotations.

Recommended:

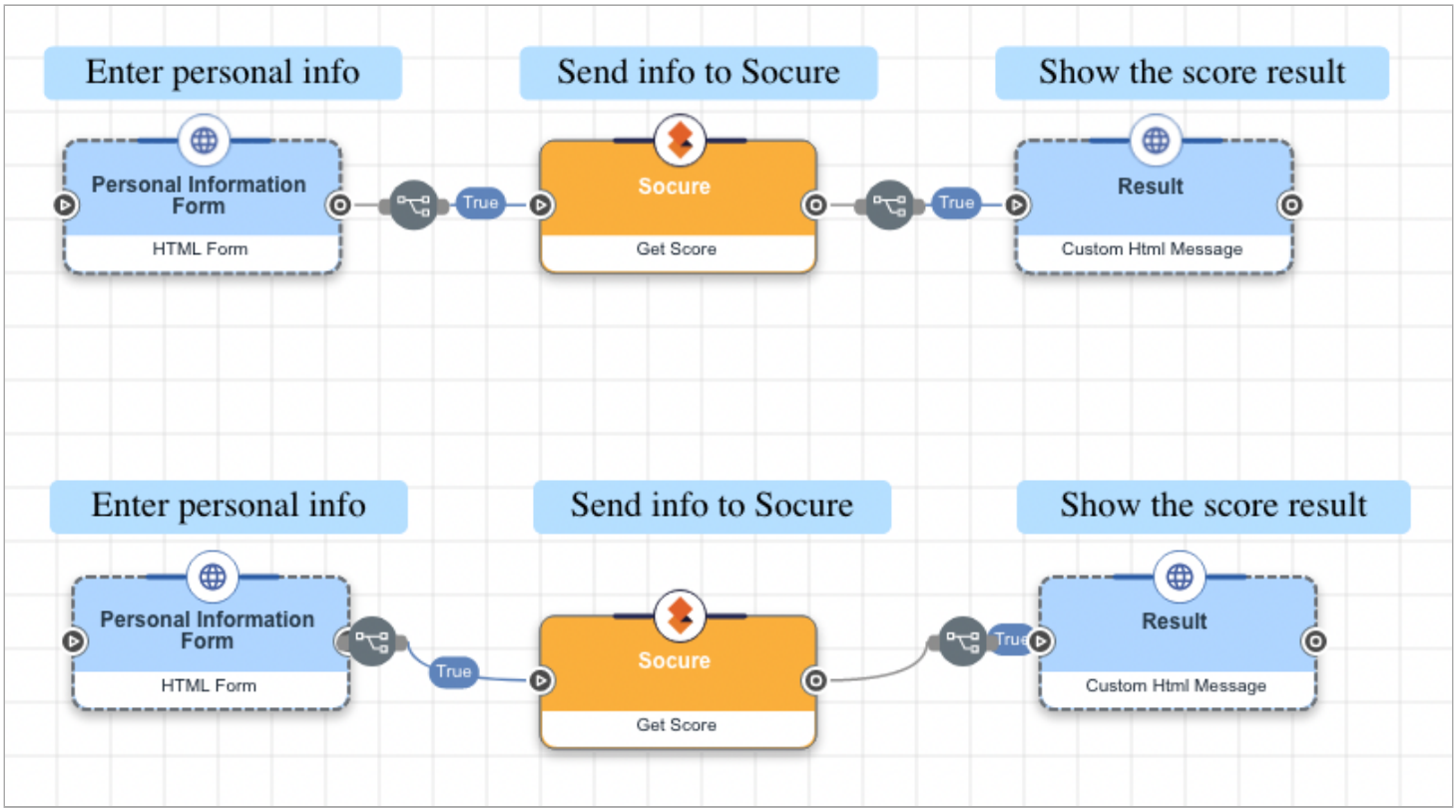


Not recommended:



Align nodes in straight lines with equal spacing

To make it easier to see the connections and overall structure, align your nodes in neat rows and columns. Avoid overlapping nodes and clustering them close together.



Use a single start node

Begin your flow with a single node that serves as a clear starting point. If you include loops in your flow, don't loop back to this start node. Instead, loop back to a later node.

> **Important**
>
> Do not begin the flow with a **Teleport Start** node. Routing to an initial **Teleport Start** node can cause flow timeout issues.

Host images outside of flows

Large images and assets should be hosted on a CDN and referenced by the flow settings using CSS, not packaged within the flow.

Use Teleport nodes for large flows

If you are creating a very large flow, you can use Teleport nodes to connect different sections of the flow. This makes the flow more readable by breaking it down into discrete sections. It can also let you reuse some sections of the flow rather than duplicating content.

> **Tip**
>
> Configure the logical operator that immediately follows a destination Teleport node to **All Triggers True**. Use a different node following each destination Teleport node; don't create flow patterns where multiple destination Teleport nodes connect directly to the same node.

Include an ID for input elements

If you're including an input element in an HTML or other user-facing node, include an ID:

```
<div>
     <label>Bar: <input id="bar" type="text" /></label>
</div>
```

You can reference the value entered by the user later in the flow using this ID. The ID must be unique across the entire flow.

Validate your flows

DaVinci includes a **Validate Flow** option that identifies errors that will break your flow as well as smaller issues that could make it harder to maintain. Use this feature regularly when developing and testing your flows, and fix the errors it identifies before you deploy the flows. Learn more in Validating a flow.

Consider flow limits

There are some limitations in flow creation, such as a maximum number of node exections and a maximum number of saved versions of each flow. Review these limitations before creating flows, as they could affect the way you create and update your flows. Learn more in Flow limits.

Consider flow complexity when designing and testing

The more complex the goal of a flow is, the more planning and testing is required to ensure that the flow works correctly. When planning a flow, bear in mind that some tasks and elements will increase the complexity and testing requirements:

- Parallel branches that run simultaneously
- Large amounts of data processing
- Looping flows
- Large flows
- API normalization

Simulate latency using the HTTP connector

If you want to simulate latency in a flow, use the **Simulate Latency** capability of the HTTP connector instead of using custom code.

Delete unused nodes

If you don't want an existing node to be part of the flow, delete it entirely instead of disconnecting or disabling it.

Delete unused flows

If you don't want to use a flow, delete it to make it easier to find relevant flows.

Branching

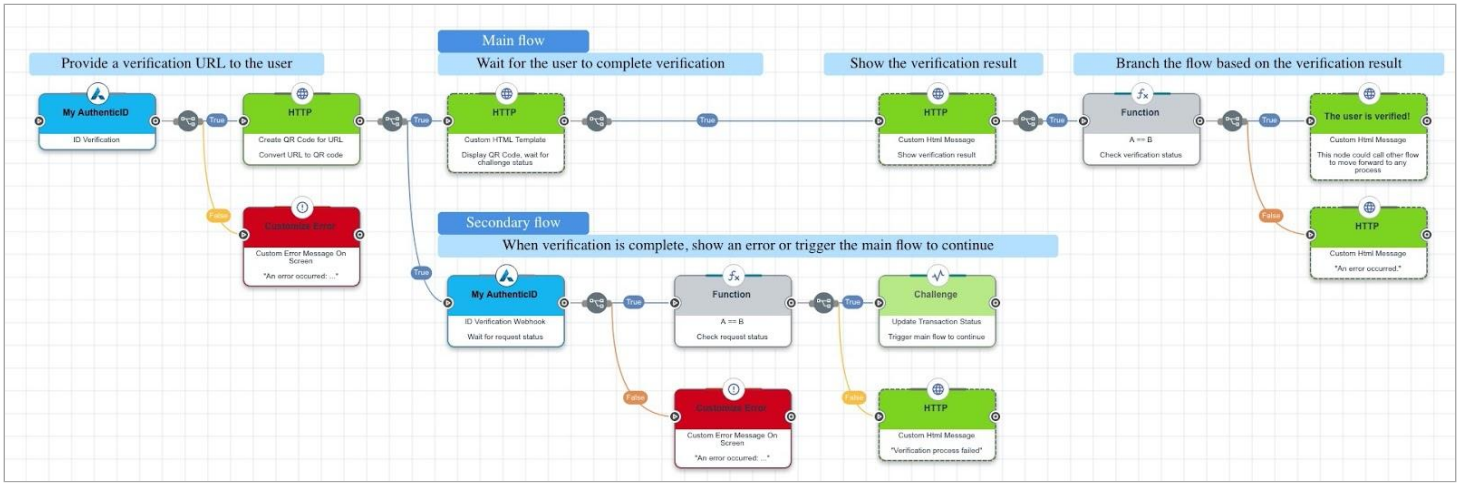
Use these best practices when creating branches within a flow.

Build the main path along the top

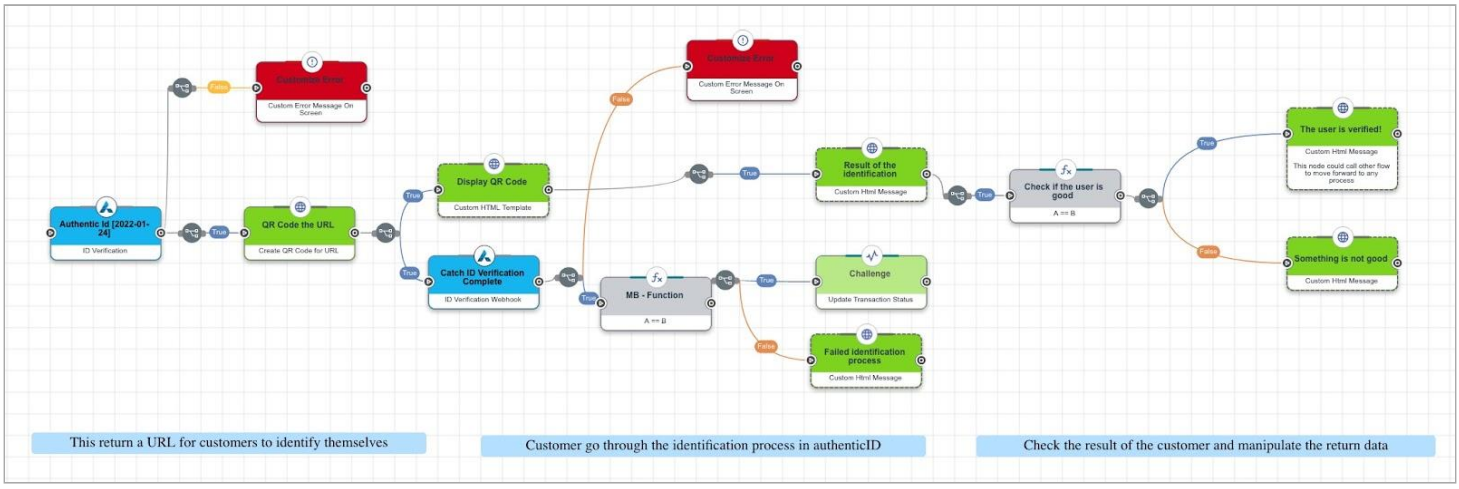
The most common success path should continue in a straight line to the right from the starting node. Alternate paths and error messages should branch downward from the main flow.

For example, consider the following flows. They are identical aside from layout and annotations:

Success path along the top (recommended):



Success path moving up and down (not recommended):

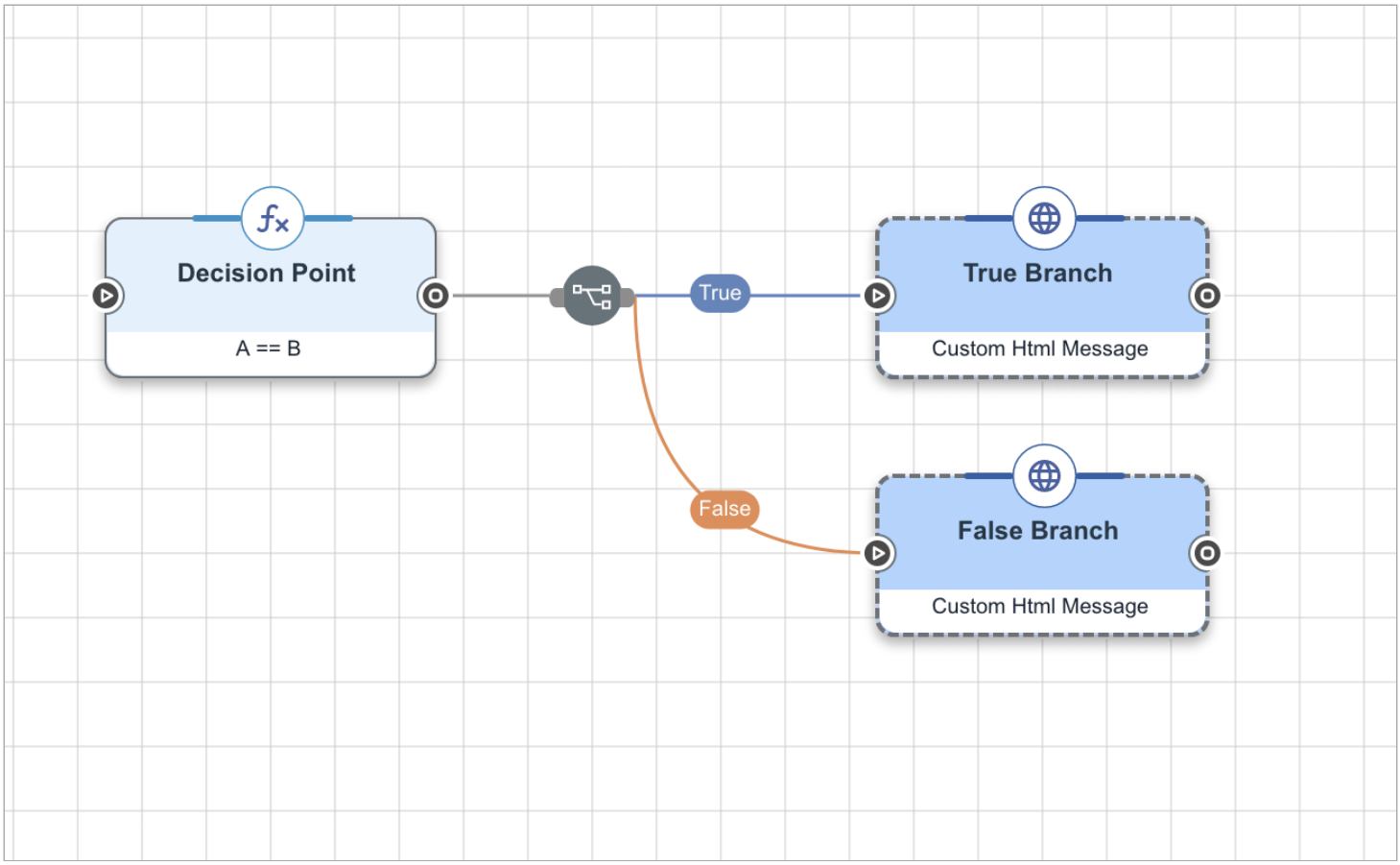


Branch flows using one logical operator per node or exit path

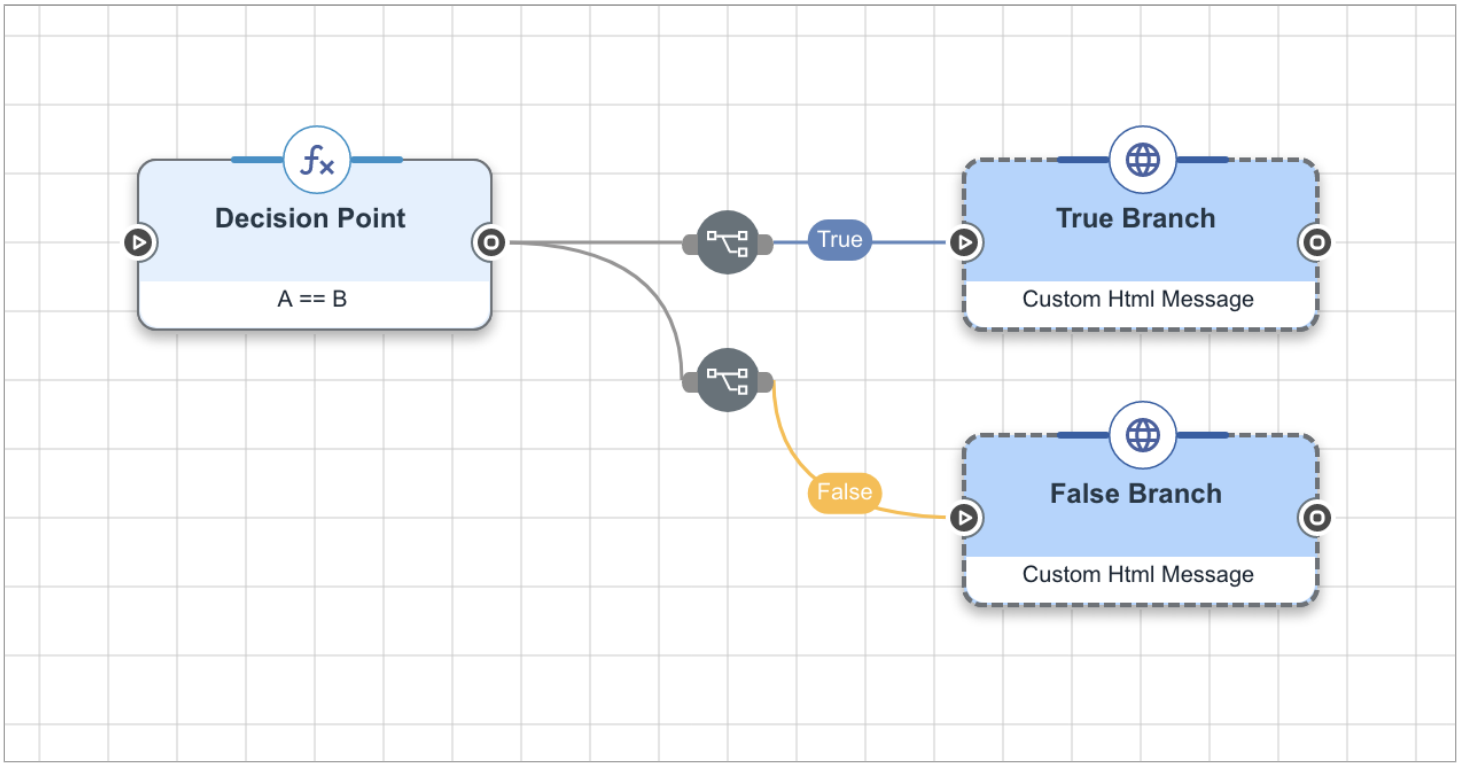
A logical operator is the circular branching point that is automatically created when you join one node to another. When you create a branch in a flow, make sure that there is exactly one logical operator per node or exit path.

If you branch after a simple node, the branches should all emerge from a single logical operator:

Branch originating from logical operator (recommended):

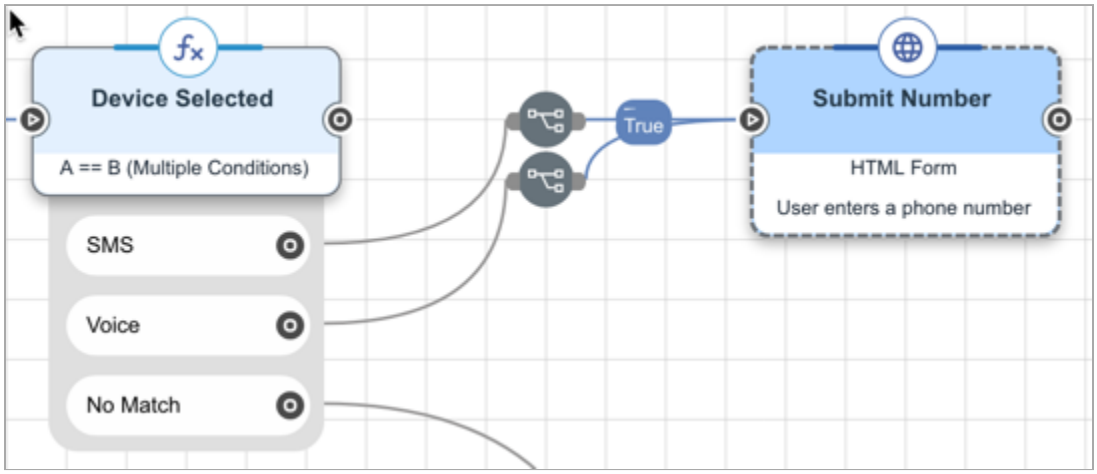


Branches originating from connector (not recommended):

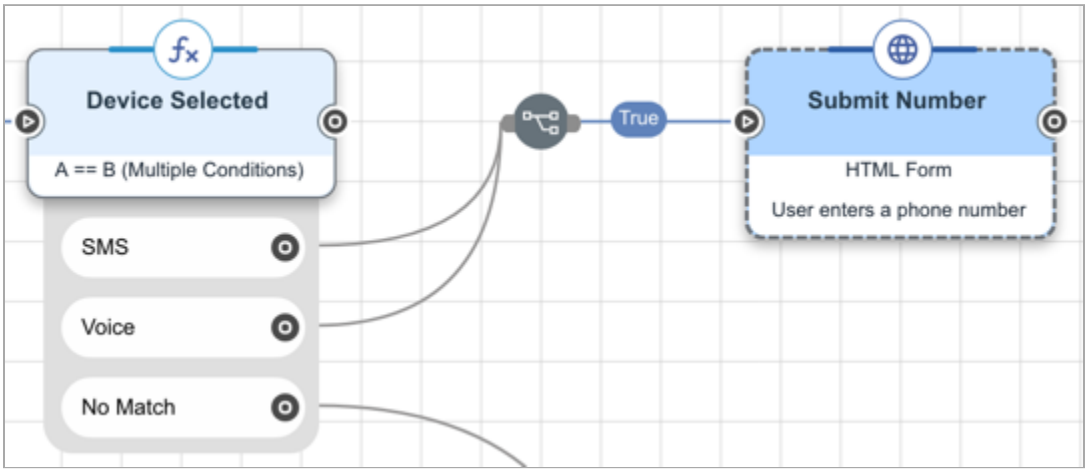


If you are branching from a node, such as a Functions connector node, that has multiple exit paths, use one logical operator for each exit path:

Branches using one logical operator per exit path (recommended):

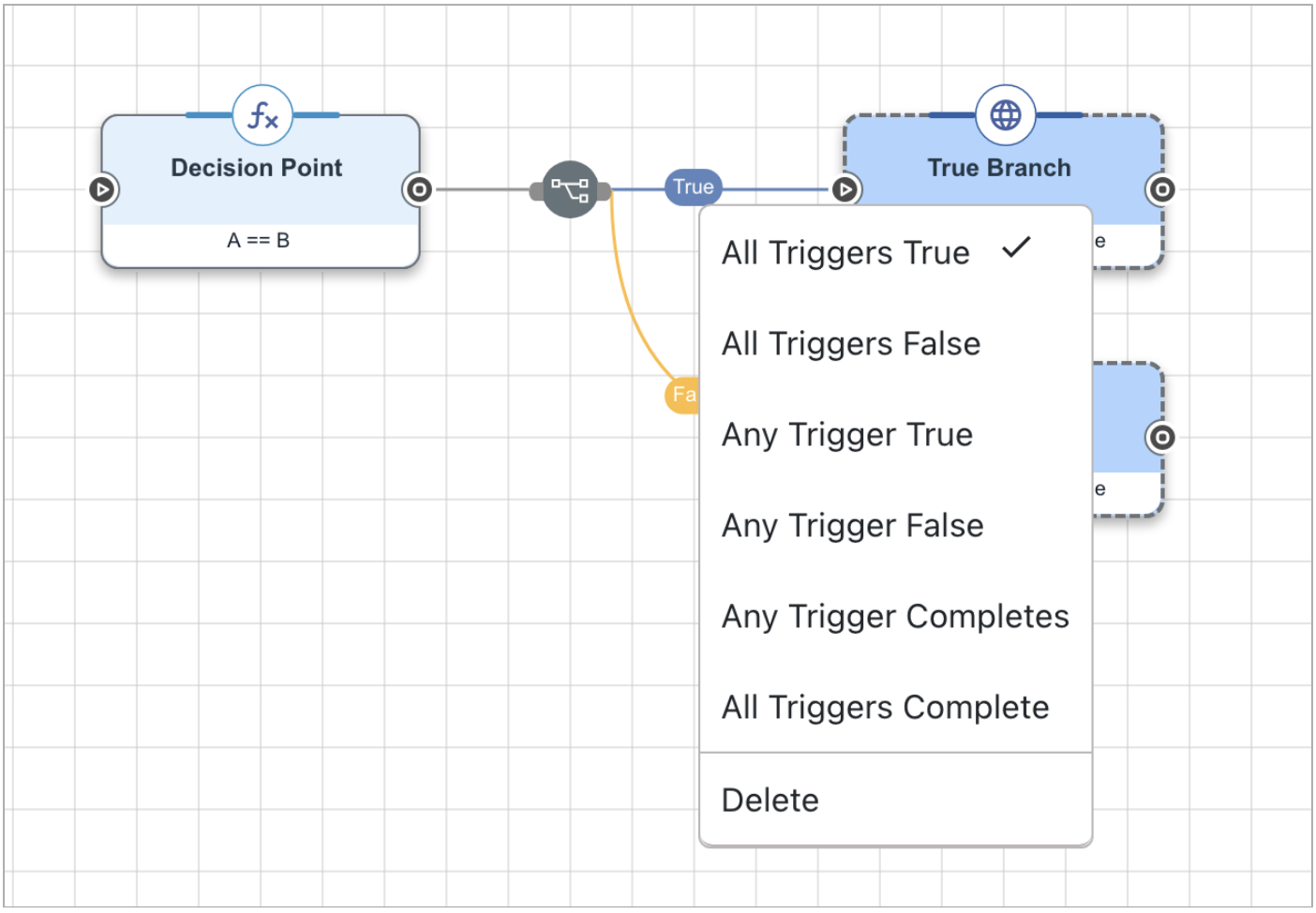


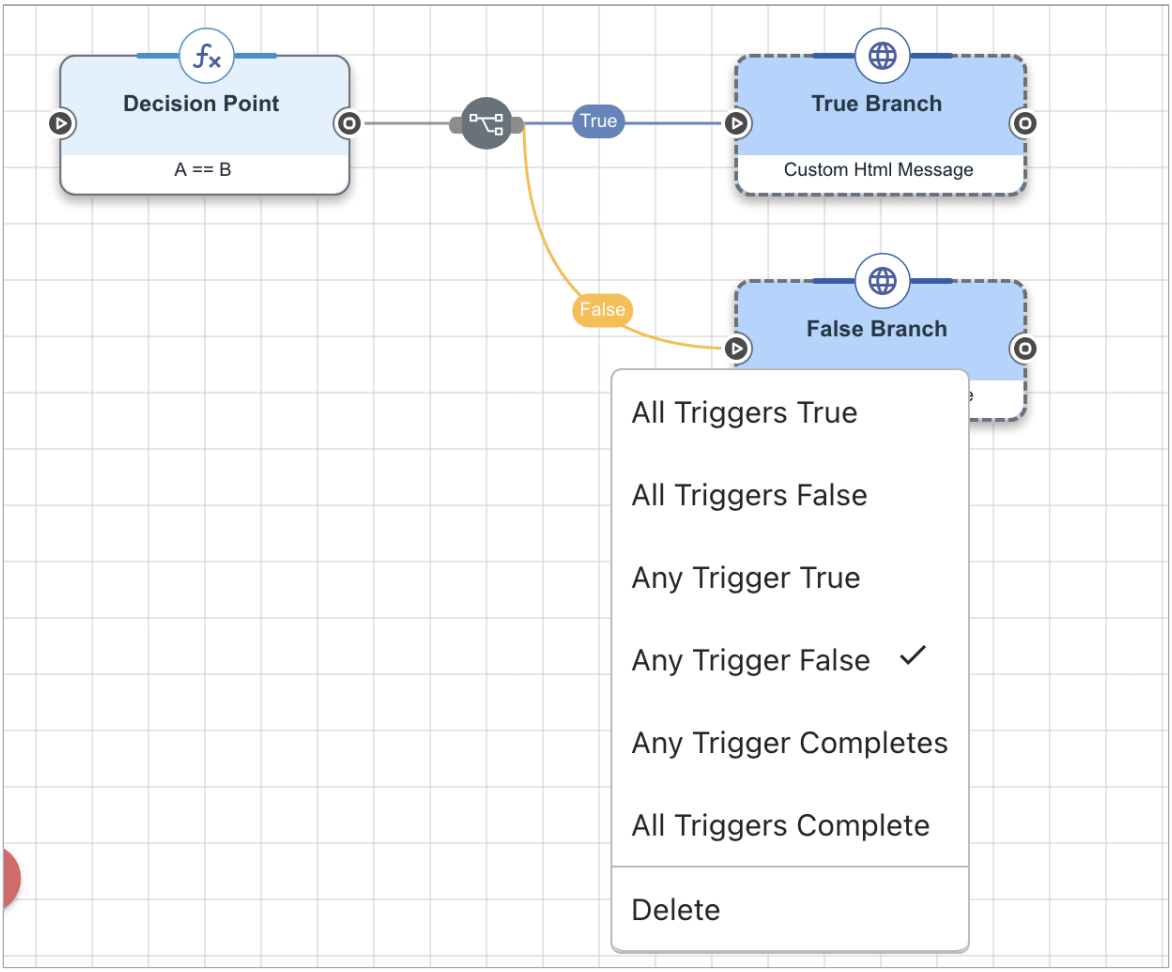
Branches using one logical operator for multiple exit paths (not recommended):



Branch using All true **and** Any false

If creating a true/false branch, you should select the **All Triggers True** option for the true branch and **Any Trigger False** for the false branch.



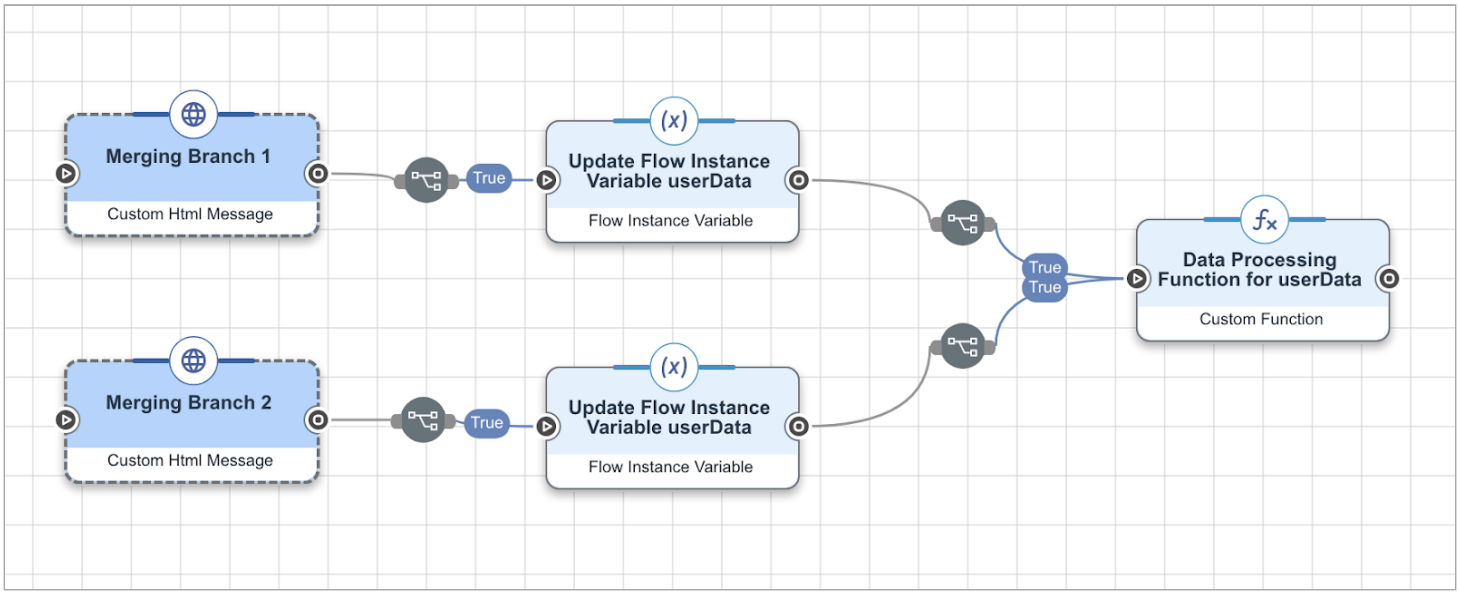


> **Note**
>
> When you duplicate multiple nodes, the connecting option might default to **All Triggers True** in the duplicated nodes. Verify that the option is correct after you duplicate nodes.

Merge branches with common data using flow instance variables

When branches that perform similar tasks converge, you can make sure you reference the data from the correct branch by setting a flow instance variable to contain the data. Nodes that come after the flow converges can reference the flow instance variable.

This solution applies where the flow branches on a choice and can follow one path or the other, then converges back together. If your flow has similar data in parallel branches that run simultaneously, you don't need a flow instance variable.

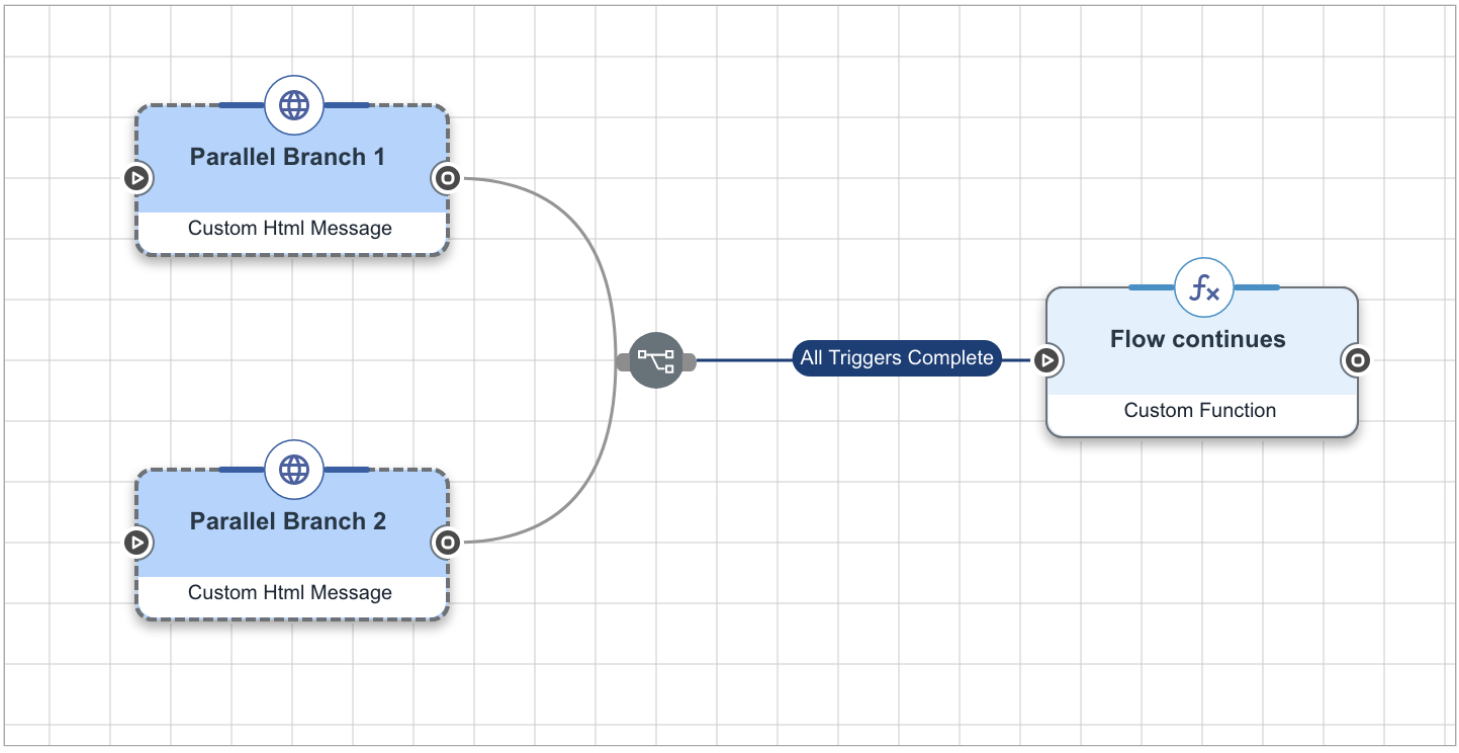


Don’t build flows with simultaneous UI paths

When branching a flow, make sure that the user can only encounter one user-facing node at a time. If the flow moves down multiple paths with UI nodes, some of these nodes will not display correctly.

Merge parallel branches using All triggers complete

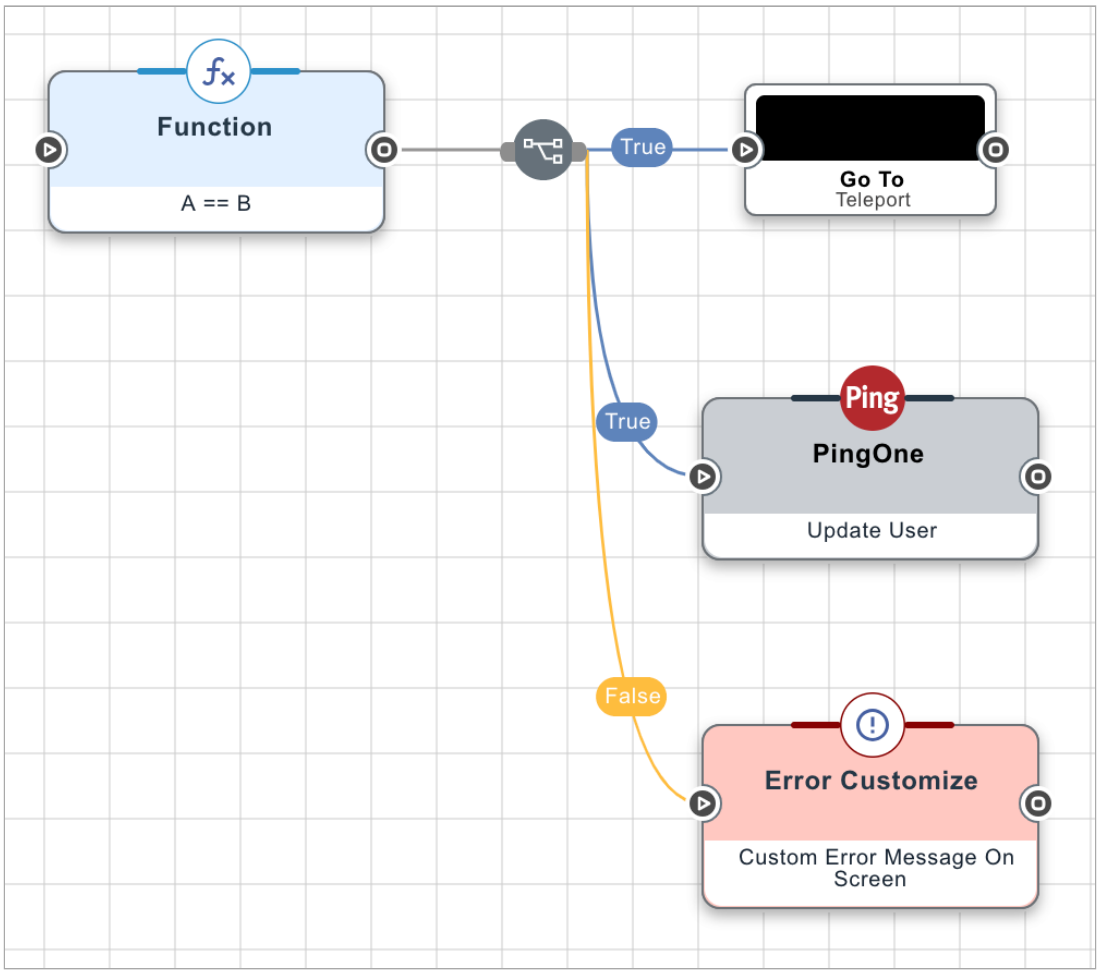
When merging branches that are executing in parallel and you want to be sure that both have completed their work, merge both branches into a connector with the **All Triggers Complete** option.



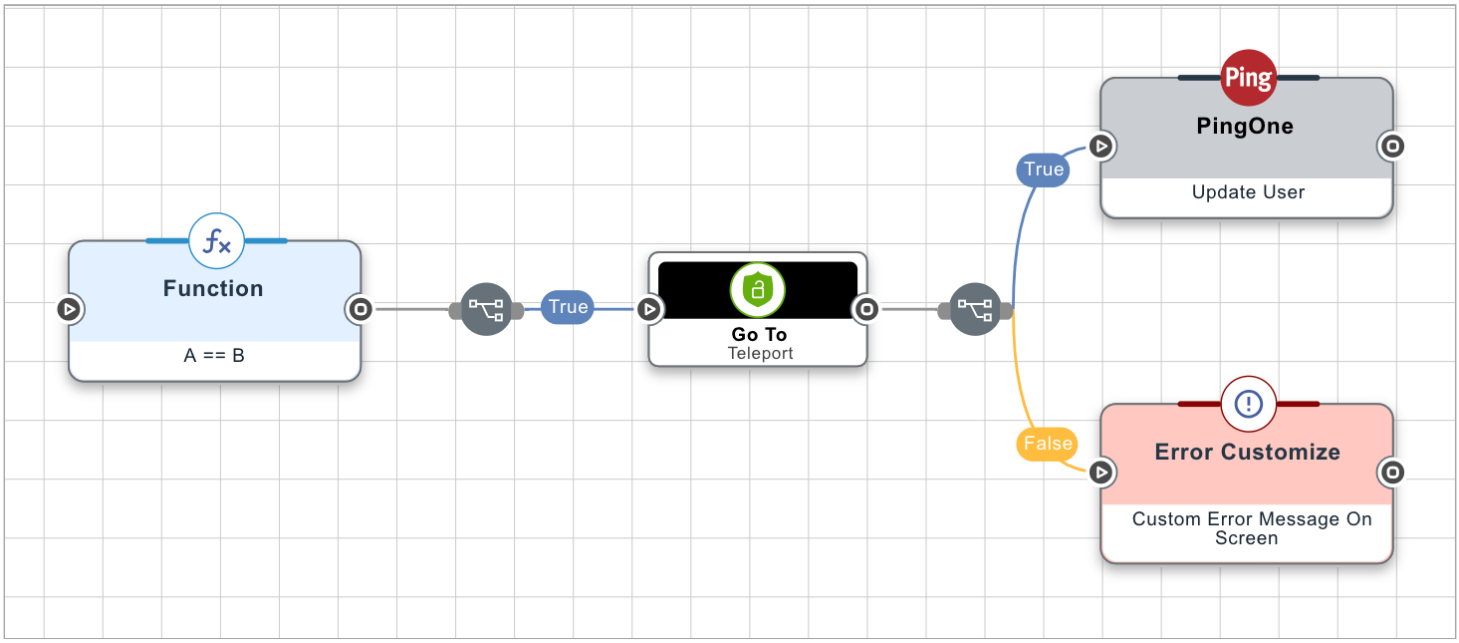
Branch before Teleport nodes

When using the Teleport connector's **Go to Start Node** or **Return to Calling Node** capabilities, branch before the node, not after. Branching after a Teleport connector that is configured to send the flow progression to a start node or calling node can skew your **Flow Analytics** data.

Branching before teleporting (supported):



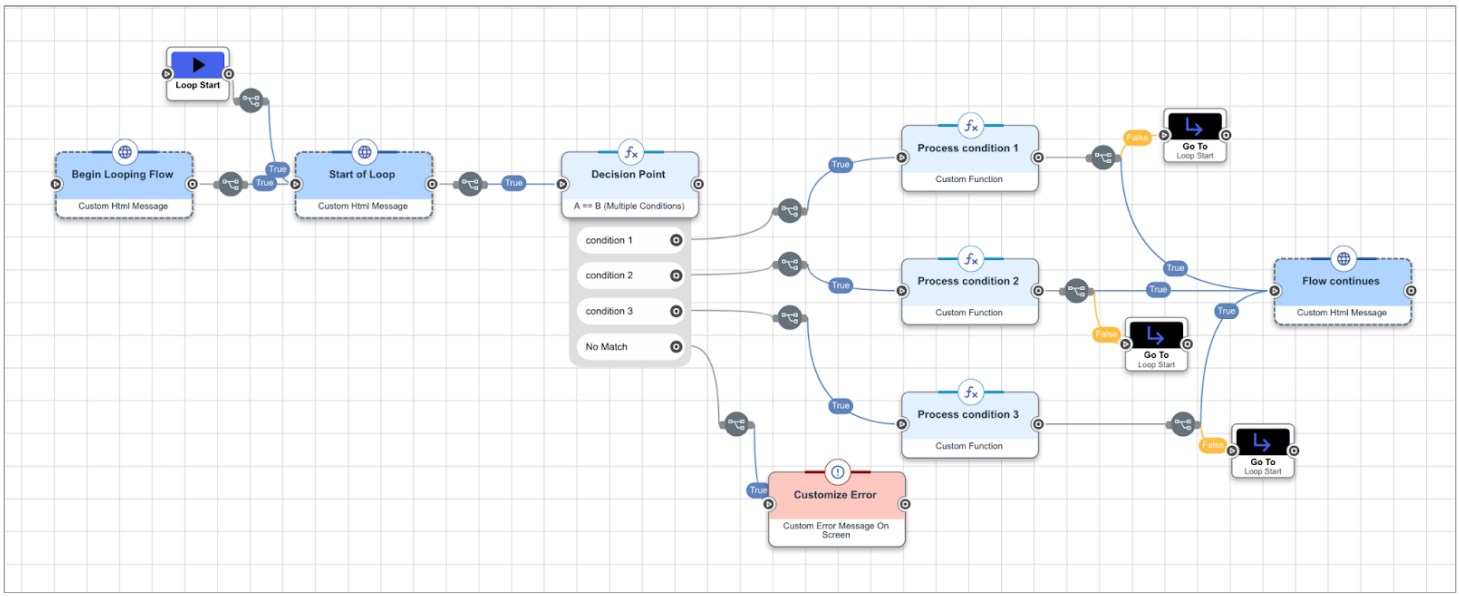
Branching after teleporting (not supported):



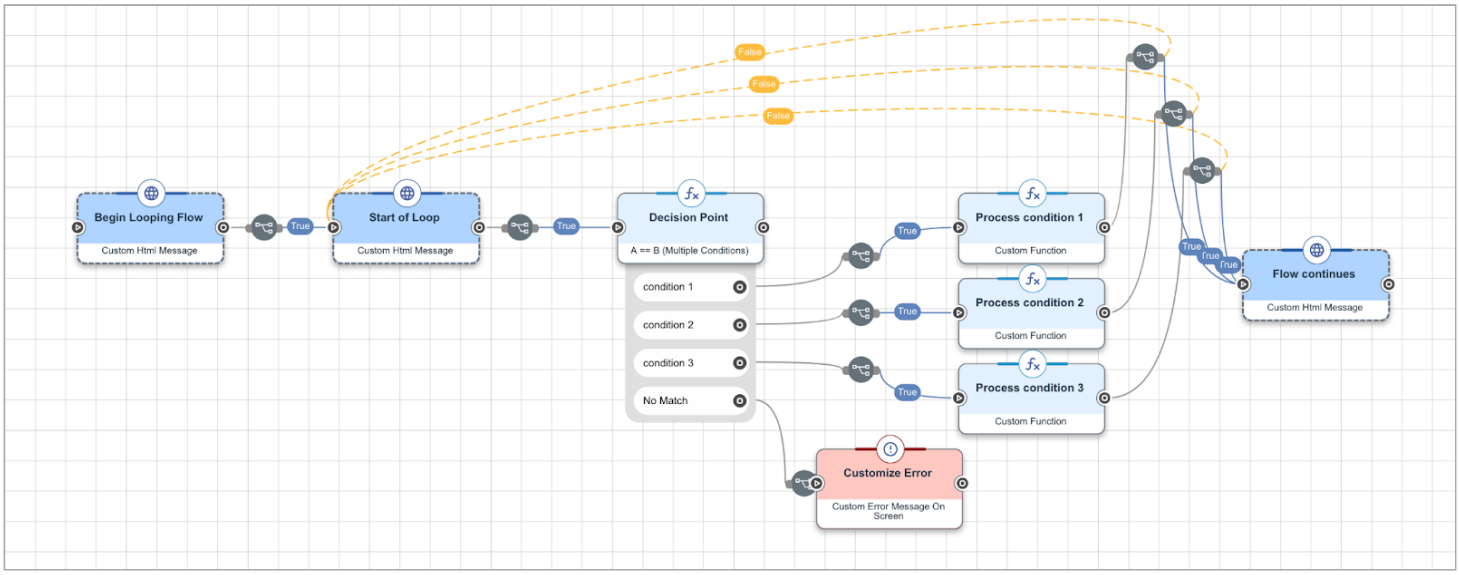
Use Teleport nodes for looping flows

When flows loop, the canvas can become cluttered with looping lines that make the flow hard to understand. To avoid this, use the Teleport connector. Define a start node at the point where the loop returns, and then use the Teleport connector to go to that start node to execute the loop.

Teleport nodes (recommended):



Looping (not recommended):



Tip

If your flow loops, you might want a field in one node to include a value from a node further to the right. When you are configuring the value for a field, you can enable the **Show all nodes** option to show values from all nodes, not just nodes to the left of the current node.

Keep Teleport destination nodes simple

Configure the logical operator that immediately follows a destination Teleport node to **All Triggers True**.

Make sure there is a one-to-one relationship between the teleport destination node and the node that follows it. You don't want to branch the flow immediately after the teleport destination node, and you should reuse teleport destination nodes rather that creating multiple destinations that lead to the same following node.

Handle all potential false branches

If a node returns a false result but does not have a false branch as an output, the flow will fail, presenting a confusing experience for the end user and a challenge for troubleshooting.

To prevent this issue, you should make sure any node that can return a false result has a false output that presents an error message.

Creating variables

To preserve variable capacity and reduce administrative overhead, follow these best practices when creating DaVinci variables.

Reuse variables when possible

Your environment can contain up to 100 variables. To limit the number of redundant variables in your company, before creating a new variable, review the **Variables** tab to see if a variable has been defined for the value that you need. If a variable exists and satisfies the flow requirements, you can reuse that variable in your flow.

When reviewing existing variables for reuse:

- Read the **Name** and **Description** fields to guide you to potential matches.
- Confirm that the **Context**, **Data Type**, and **Value** parameters work for your flow design.
- Click **Edit** on a variable to review its **Mutable**, **Min**, and **Max** configurations.

> **Tip**
>
> Flow instance variables are always safe to reuse.

Standardize your variable names

To help flow builders quickly identify which variables pertain to their desired values, standardize your variable names and description requirements, and then communicate these standards to all DaVinci stakeholders. You can define standards for individual variables or variable categories.

You can find examples of standardized variable names in Common flow variables.

Choose an appropriate variable context

To allow a variable to be used in as many flows as possible, select the most general context that applies to the purpose of the variable. Variable contexts apply as follows:

Company Context

With this context, the variable has one definition for all flows and users. Consider selecting **Company Context** for variables that represent your company name or a company-wide ID.

User Context

With this context, the variable has a separate value for each user. Consider selecting **User Context** for variables that represent user-specific data, such as a user ID or status.

Flow Instance Context

With this context, the variable's value can be set upon execution of a flow that contains the variable, and that value is unique to the containing flow. The variable can also inherit a value from the **Variables** tab, if no value is set by the execution of the its containing flow.

Consider selecting **Flow Instance Context** for variables that represent information that might change from one flow execution to the next (even for the same user), such as a time value.

Flow Context

With this context, the variable is tied to a specific flow and has a persistent value. This value is set by the latest of either an execution of its containing flow or an update to the **Variables** tab. Consider selecting **Flow Context** for variables that represent values that are common to all users within a flow, such as a task ID or task-related information.

For more information on variable context, see Variables.

Use secret variables for sensitive information

Some values, such as API keys, client secrets, and access tokens, are required for some flows to function but should not be readily visible. You should store these values in secret variables.

Secret variables are a type of company context variable. Their values can be used in the HTTP connector's Make REST API call capability, in the Headers and Body Parameters sections. Their values are not visible in logs, analytics, API queries, flow JSON files, or to other DaVinci administrators. Administrators can replace the value for a secret variable, but they cannot view the current value after it's been saved. Saving sensitive values in secret variables makes them more secure and simplifies administration by putting all of the values in one place rather than in multiple connector configurations.

Learn more in Variables and Adding a variable.

Collaborating

Use these best practices when collaborating with other flow builders to create or update a flow.

Use subflows

If you can break a large flow into separate, smaller flows (subflows), each flow builder can work independently. See the Subflows section for best practices.

Rename flows to show working status

Rename the flow with your initials while you are working in the flow, and change the name back to the original name when you are finished. This will let other flow builders know to wait until you are finished with your changes.

Subflows

Use these best practices when creating and using subflows (flows invoked by other flows).

A subflow is a normal flow that does not represent a complete identity orchestration solution or user experience. Instead, it's designed to be used as part of a larger parent flow. For example, a complex sign-on flow might reference subflows to handle password reset requests and user registration. Subflows make flow maintenance and reuse easier.

> **Important**
>
> To invoke a subflow, you must use the Flow Conductor connector.

There are two types of subflows.

UI subflows

Subflows that include UI components (for example, an HTTP connector), such as:

- Magic link flow
- Login/authentication

Non-UI subflows

Subflows that don't include UI components. These subflows perform backend functions such as:

- Common API calls (for example, create a user or reset a password)
- Computational functions

Best practices for creating subflows

Use these best practices when creating a subflow.

Begin the subflow name with Subflow

When naming a new subflow, begin the name with **Subflow**.

Don't use CSS, even for UI subflows

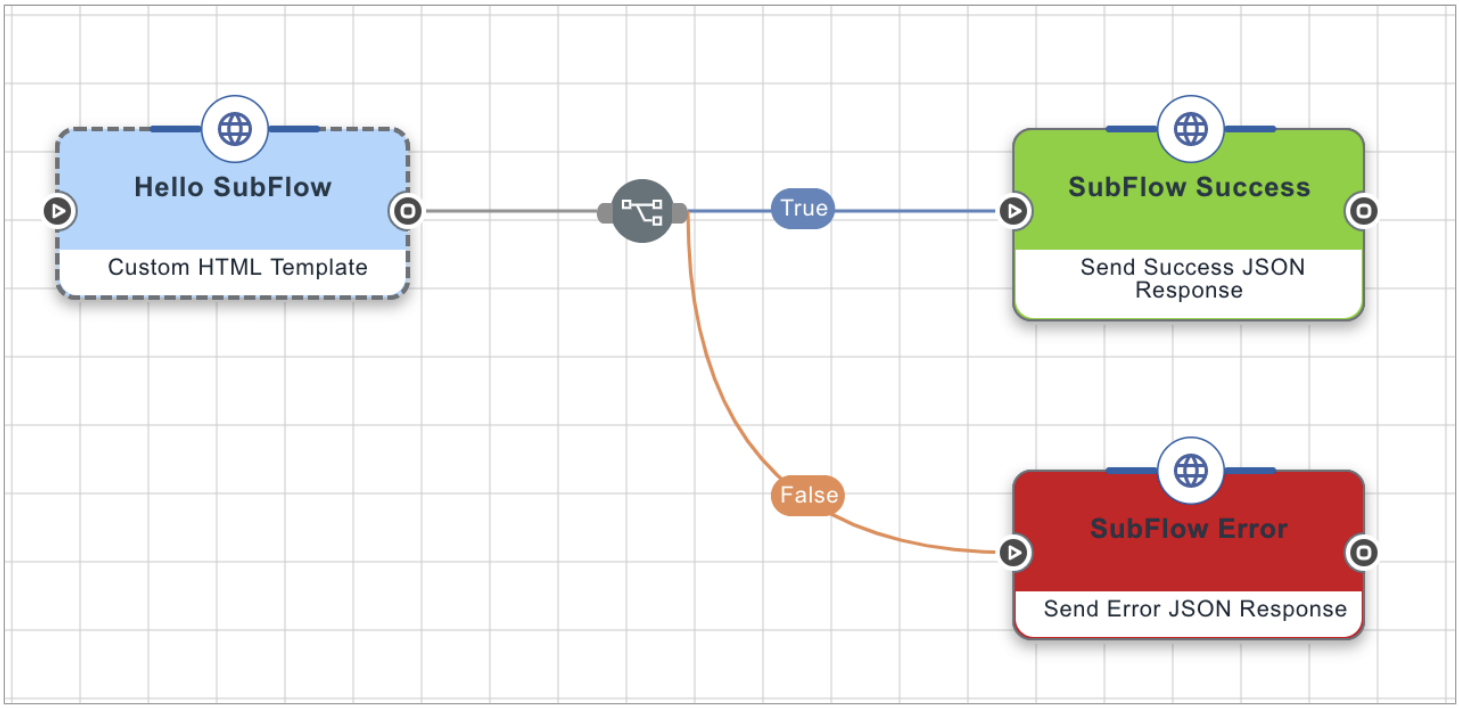
Don't include CSS in subflows, even UI subflows. The subflow inherits the CSS styling from the parent flow, and controlling the CSS from that top-level flow is preferable.

Don't configure subflows as PingOne flows

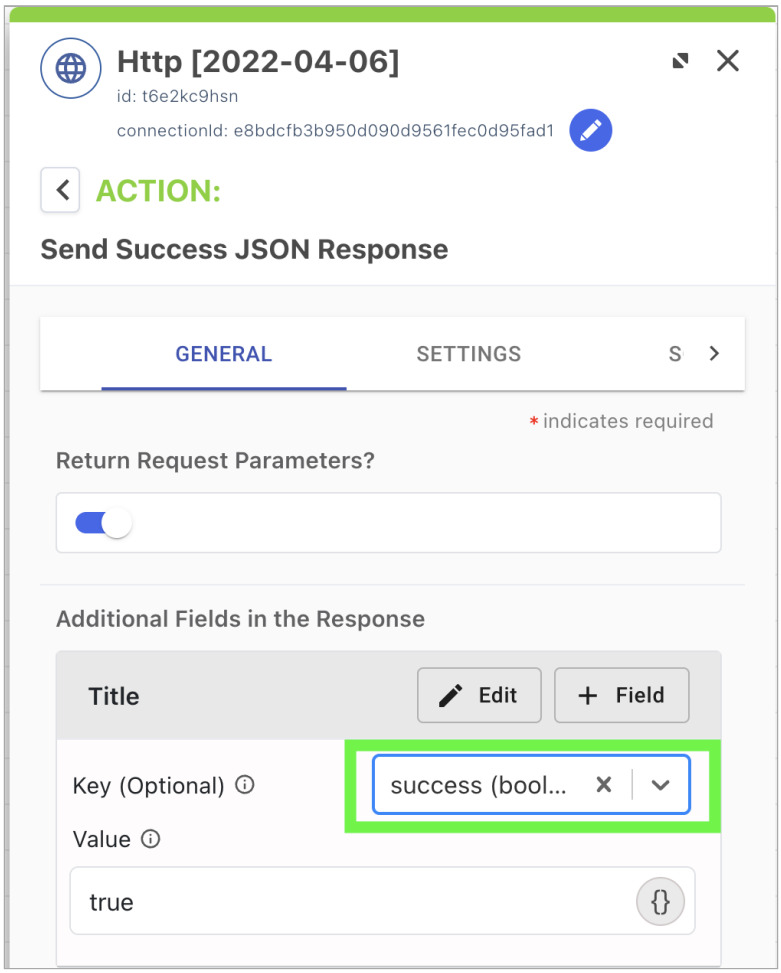
For subflows that connect to PingOne main flows, only configure the main flow as a PingOne flow, not the subflow. For more information, see Editing flow settings.

Add color-coded HTTP connectors for success and failure

Use HTTP Connector nodes to send a JSON success or error response at the conclusion of the subflow. Give clear titles to these nodes and color them according to their purposes.



Add a boolean value to each JSON Response node, specifying whether the subflow was successful.



Best practices for using subflows

Follow these best practices when using subflows.

Send a JSON Response for the subflow's end state

If the subflow completes successfully, end the subflow with an HTTP node set to the **Send Success JSON Response** capability. In the parent flow, the flow conductor node that launched the subflow then evaluates to true.

If the subflow completes unsuccessfully, end the subflow with an HTTP node set to the **Send Error JSON Response** capability. In the parent flow, the flow conductor node that launched the subflow then evaluates to false.

The **Send Custom JSON Response** capability is not recommended for use in subflows. If a subflow ends with a node using this capability, the flow conductor node that launched that subflow always evaluates to true. If you plan to use a custom JSON response to conclude a subflow and you need to return a success or error condition, use a variable to store the condition, then use an evaluation node in the parent flow if you need to make branching decisions based on the variable's value.

For example, to use a custom variable in that HTTP node to indicate that the use case is an error response, use `IS_MFA_Required_Response=false`.

Use styling rules from the parent flow

Avoid defining CSS in a subflow. Instead, set the styles in the flow settings of the parent flow. You can override the flow-level CSS by defining CSS for individual nodes if you need those nodes to have a different appearance.

Use the appropriate capability for the subflow

When you invoke a subflow, use the appropriate capacity for the type of subflow:

- Use the **Invoke UI Subflow** capability if the subflow contains user interface nodes. Verify that every possible user path includes at least one user interface node.
- Use the **Invoke Subflow** capability if the subflow doesn't contain user interface nodes.

Minimize the number of layers (subflows calling other subflows)

Because each subflow has a performance cost, avoid solutions that require subflows to reference other subflows. Don't exceed five levels of depth. Instead, design a single parent flow that references multiple subflows in sequence.

Avoid multiple branches leading to a single subflow

If more than one branch leads to a single subflow node, the subflow might not consistently launch the correct number of times. If each branch must conclude with a subflow invocation, include a separate node for each invocation.

Don't loop subflows

If your flow invokes a subflow, make sure that the subflow doesn't invoke the parent flow.

Best practices for data sharing

Follow these best practices when passing data between a flow and a subflow.

Passing data to a subflow

Add an input schema to the subflow to pass information from the parent flow to the subflow.

To see the input schema for the subflow, open the subflow, then click **Input Schema** in the upper-right corner of the flow canvas. You can add any number of required or optional properties to the subflow's input schema.

We recommend using a consistent format, such as camel case, for naming input schema variables. For example, `userName`, `emailAddress`, and so on.

If the input variable is required for the execution of the flow, mark it as **Required**. Alternatively, you can select **Require all defined parameters** to require all parameters.

In the subflow, you can reference the input schema parameters as `{{global.parameters.paramName}}`, where `paramName` is the name used in the input schema.

In the parent flow, the **Flow Conductor** node that launches the subflow includes fields for each parameter in the input schema. Map values from the parent flow to each of the subflow's input schema parameters.

Returning data from a subflow

Add an output schema to the subflow to pass information back to the parent flow.

When you configure the **Send Success JSON Response** and **Send Error JSON Response** nodes at the end of the subflow, you can include one or more fields in the response. By adding these fields to the subflow's exit schema, you make them available in the parent flow after the subflow completes.

Click ⋮ > **Output Schema** to open the output schema, then add one or more parameters to the properties section of the output code block. For example, this output schema outputs the errorMessage and selectedDeviceOtpEnabled properties:

```
{
  "output": {
    "type": "object",
    "additionalProperties":true,
    "properties": {
      "errorMessage": {
        "type": "string",
        "displayName": "Error Message",
        "preferredControlType": "textField",
        "enableParameters": true,
        "propertyName": "errorMessage"
      },
      "selectedDeviceOtpEnabled": {
        "type":"bool"
      }
    }
  }
}
```

In the parent flow, you can reference parameters included in the output schema using the structure `{{local.nodeId.payload.output.property}}`, where `nodeId` is the node ID of the **Flow Conductor** node that launched the subflow and `property` is the property name. You can also find these parameters in the `{}` menu.

Debugging and analytics

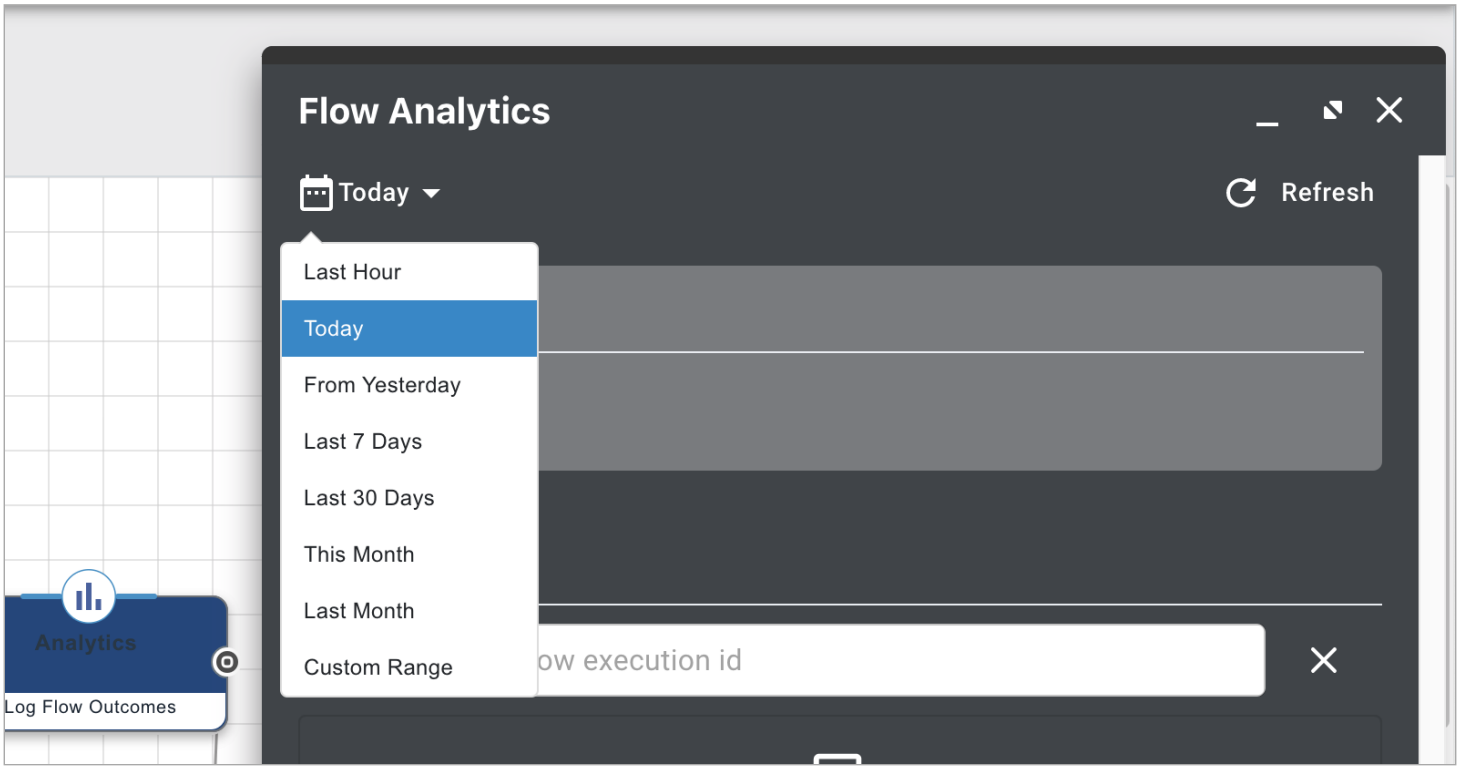
Use these best practices when troubleshooting issues with flows and reviewing flow analytics data.

Review the flow execution log

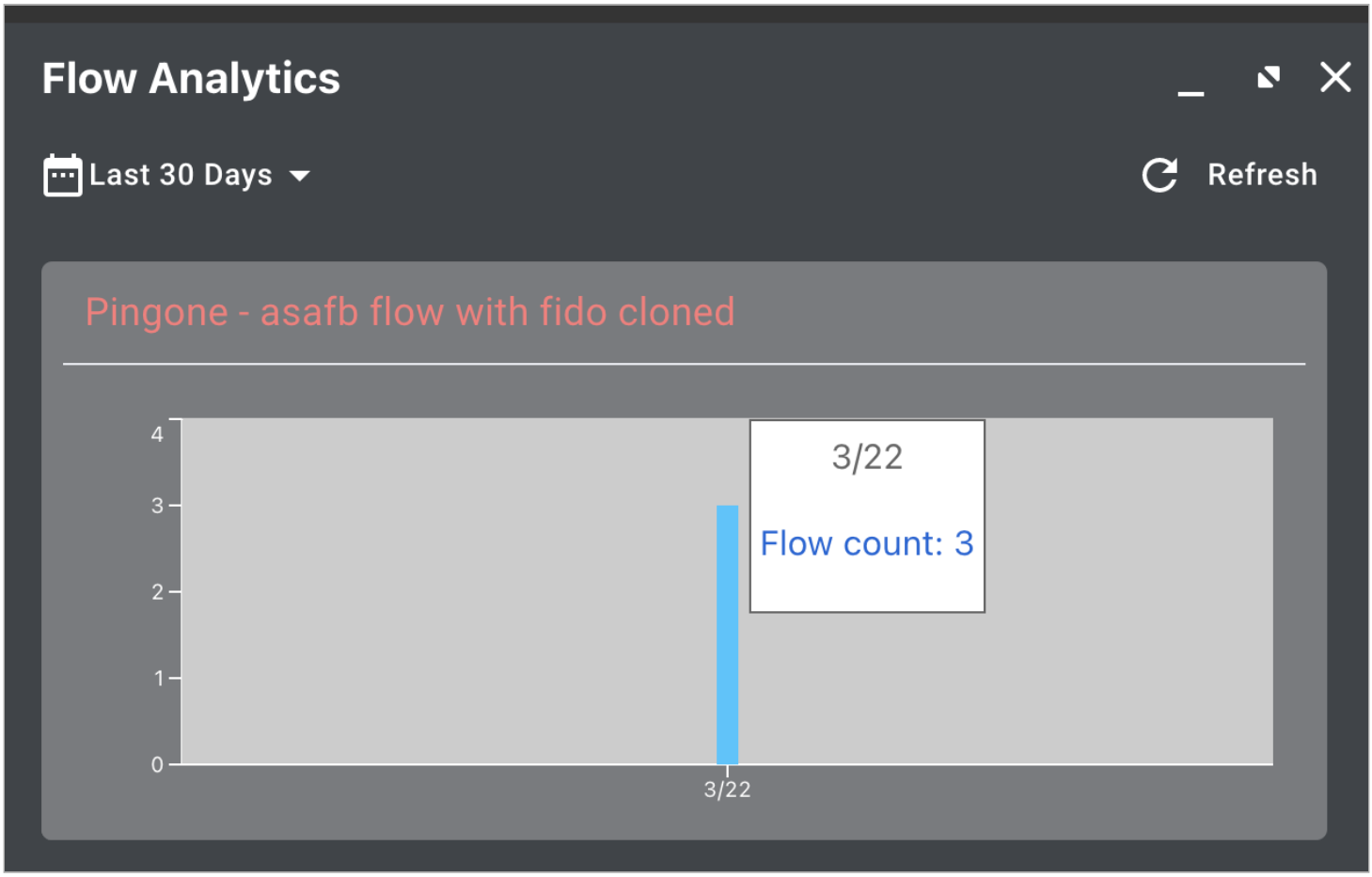
Open your flow in PingOne DaVinci.

In the lower left corner of the flow editor, click **Analytics**.

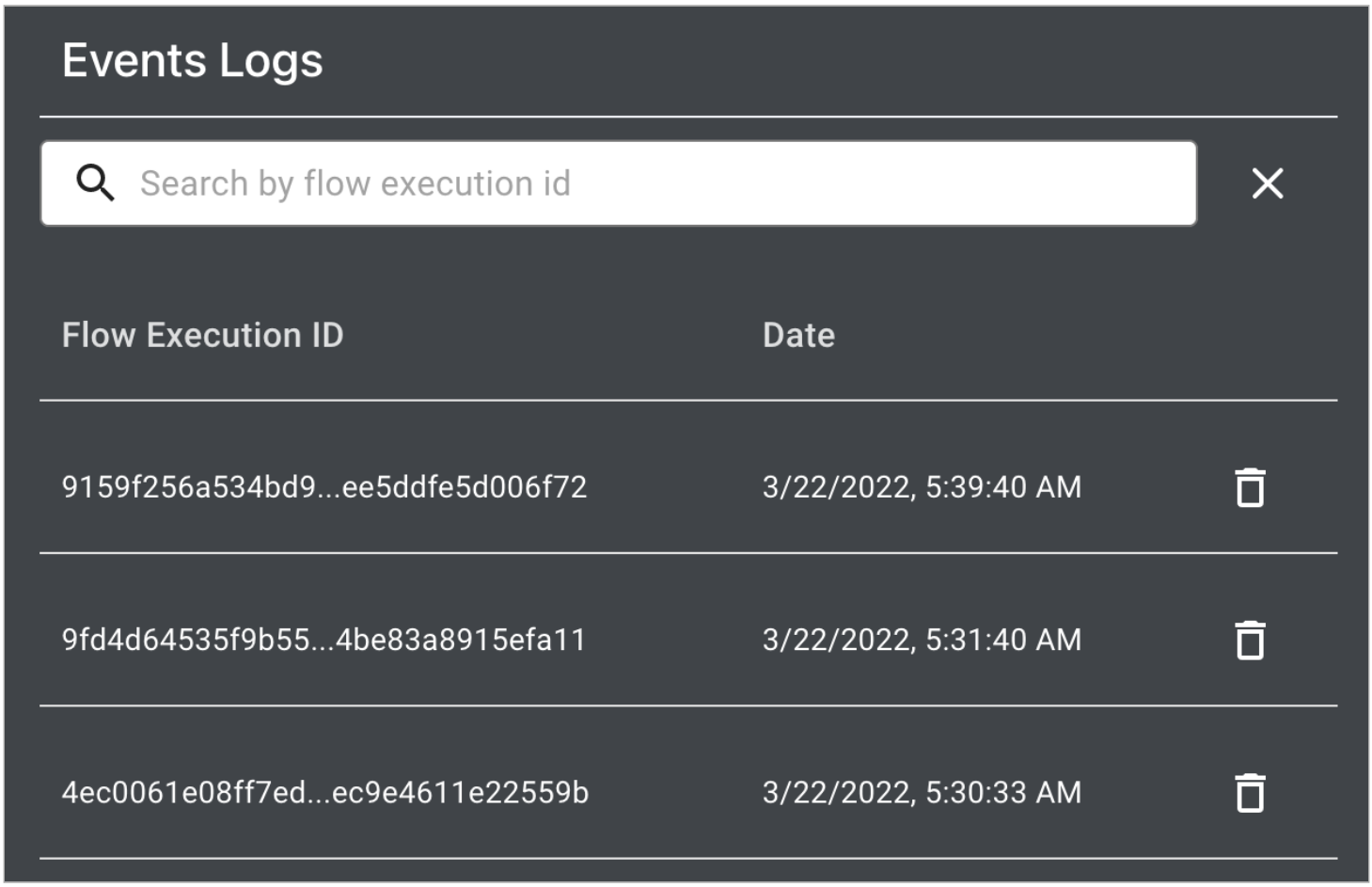
This opens the **Flow Analytics** window. You can change the date range for the executions of your flow. You can also click **Refresh** to load information for your latest flow executions.



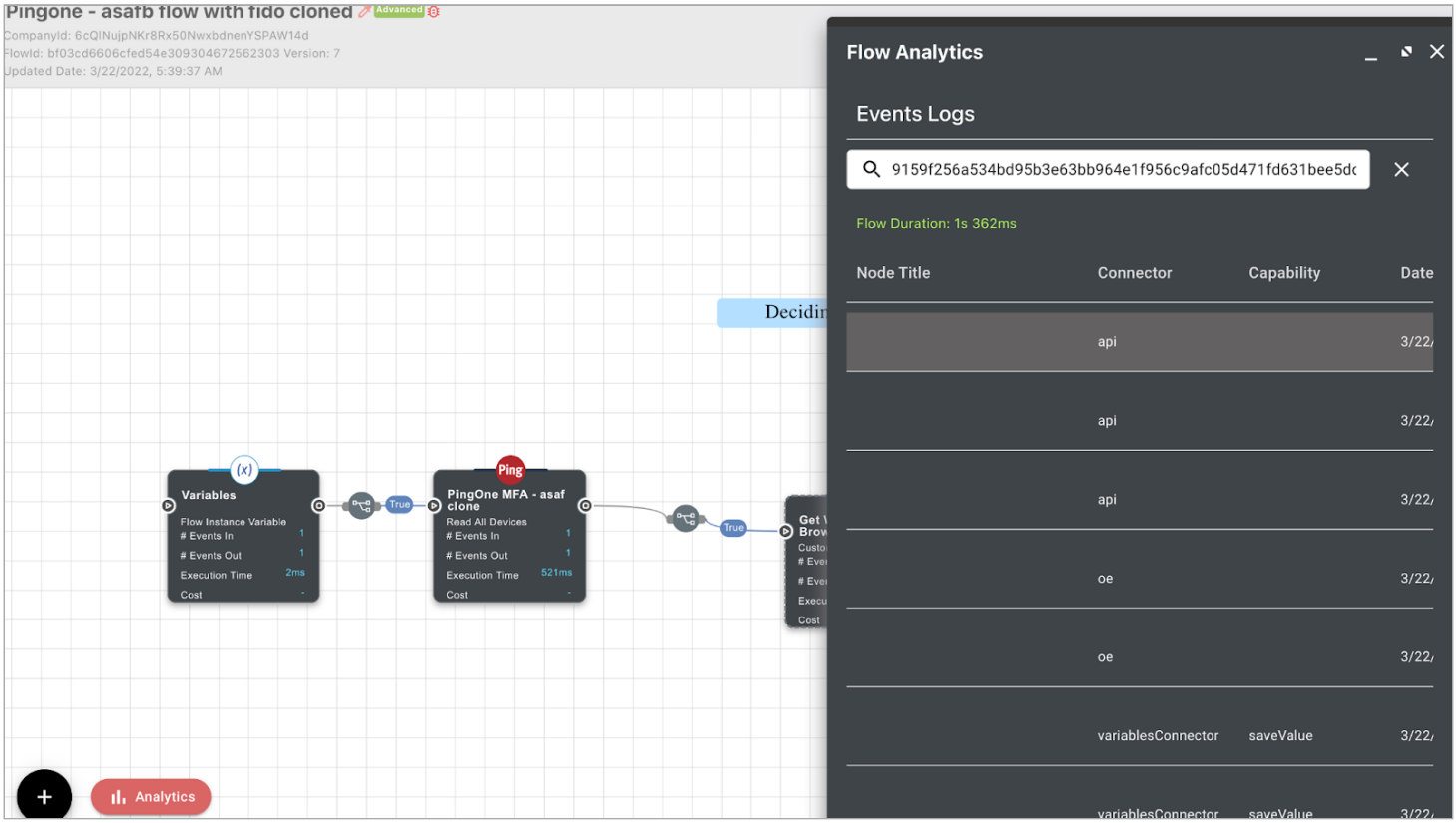
Hover over the graph to see the flow count for the selected timeframe.



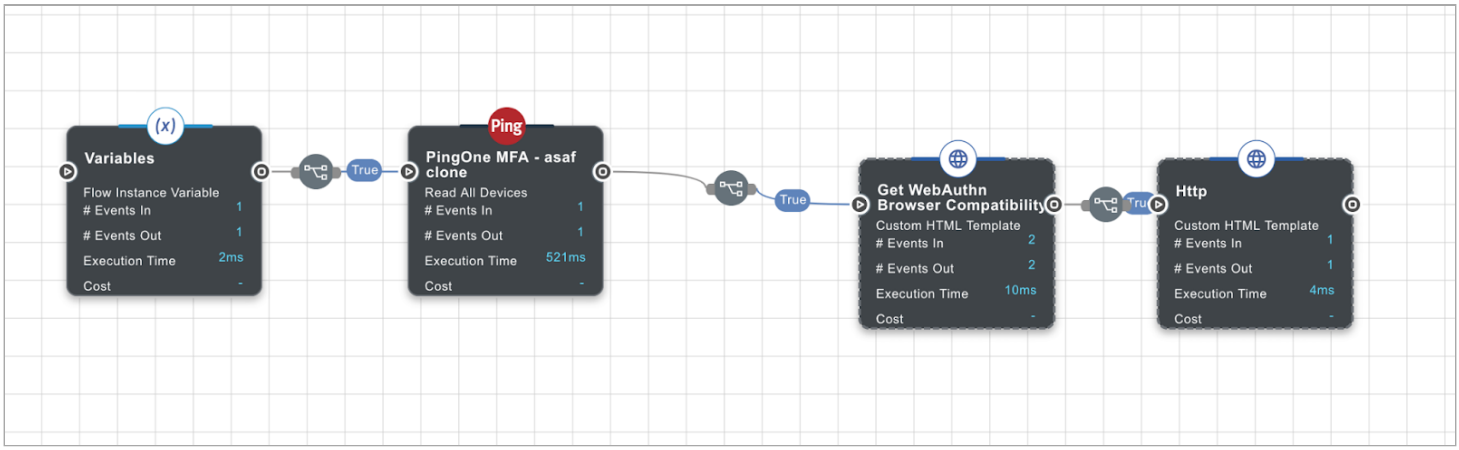
In the **Event Logs** section, you can search for a specific flow execution ID or select a flow execution from the list to see its details. Flow execution is displayed in chronological order with the most recent on top.



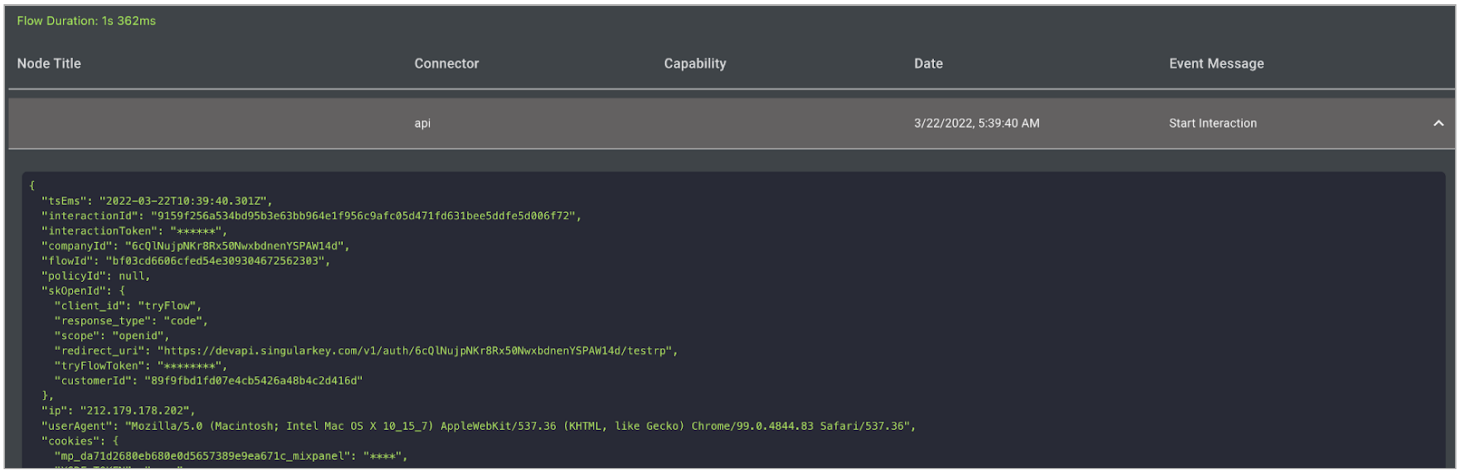
After you select a flow execution, you can see **Flow Duration** and connector details such as **Node Title**, **Connector**, and **Capability**. Expand the view or scroll to the right to see the **Date** and **Event Message** columns.



In the canvas view, the nodes show analytic information for the flow execution you selected. Hover over a node to see the related events in the **Flow Analytics** window.



Click an event in the log to expand the JSON request or response for the event. This allows you to see the information that passed through the flow, including error codes and messages.

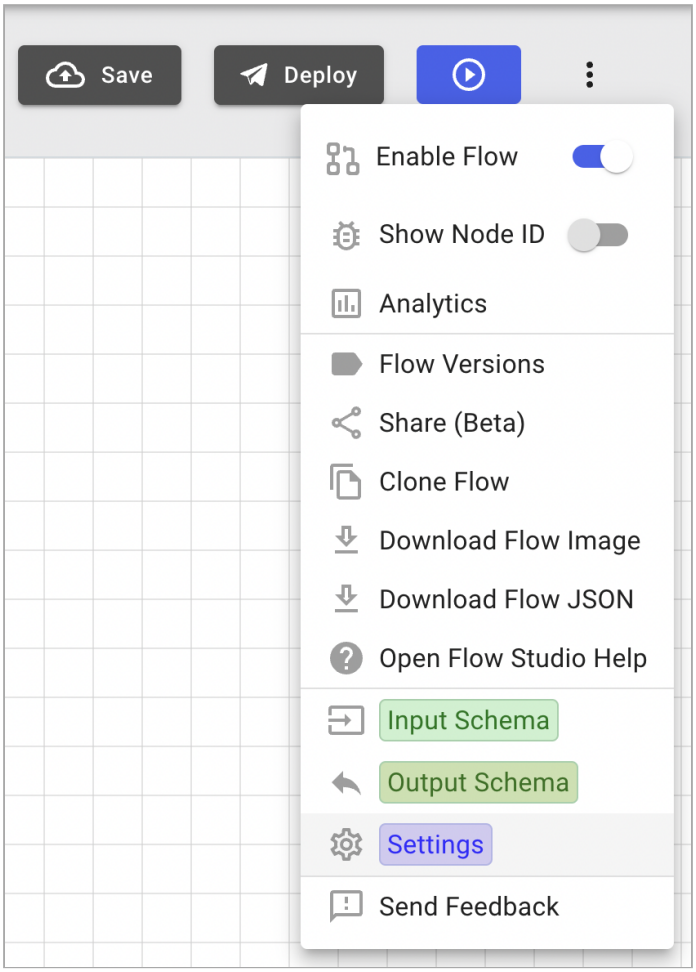


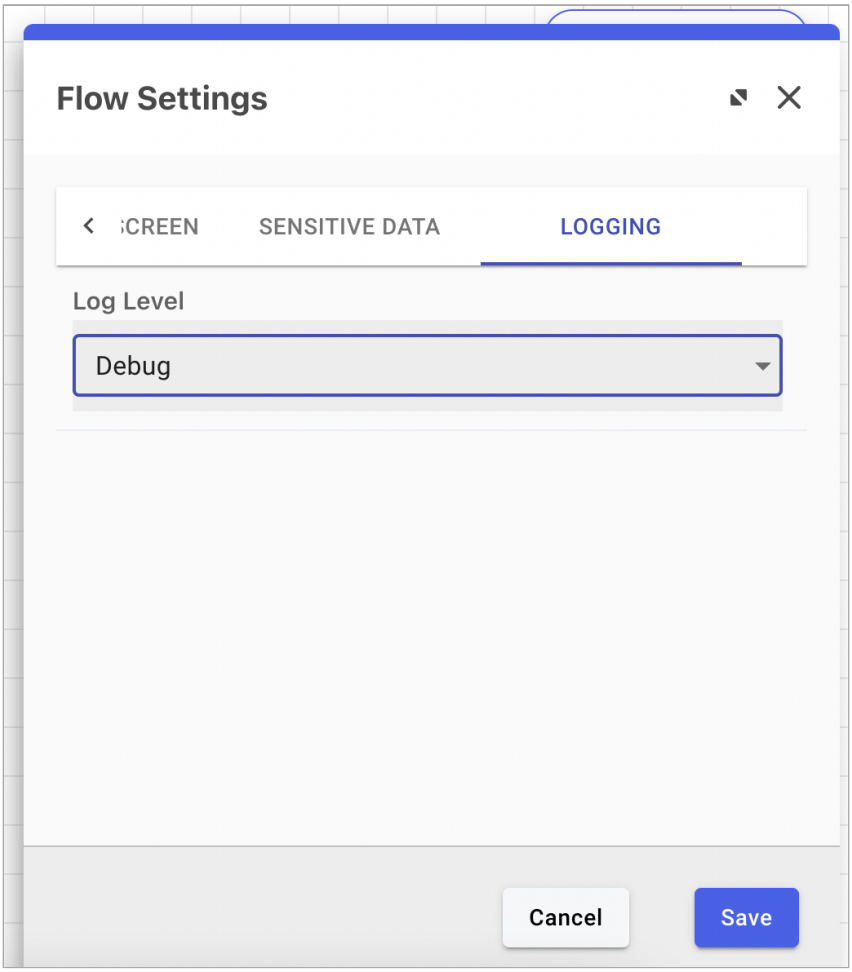
If there are any errors, you might see a **Send Error Response** that will show you the error encountered.

Use debug mode to view additional details

Use the **Debug Mode** flow setting to capture additional details in the **Flow Analytics** log. Enabling debug mode provides additional insight into the properties, parameters, and connector inputs between nodes in the flow.

In the upper-right corner of the flow editor, click the **More Options (⋮)** icon and select **Flow Settings**. On the **Logging** tab, in the **Log Level** list, select a logging level. **Debug** captures the most detailed information.





When you are done analyzing the flow, reset the log level to **Info** to improve performance.

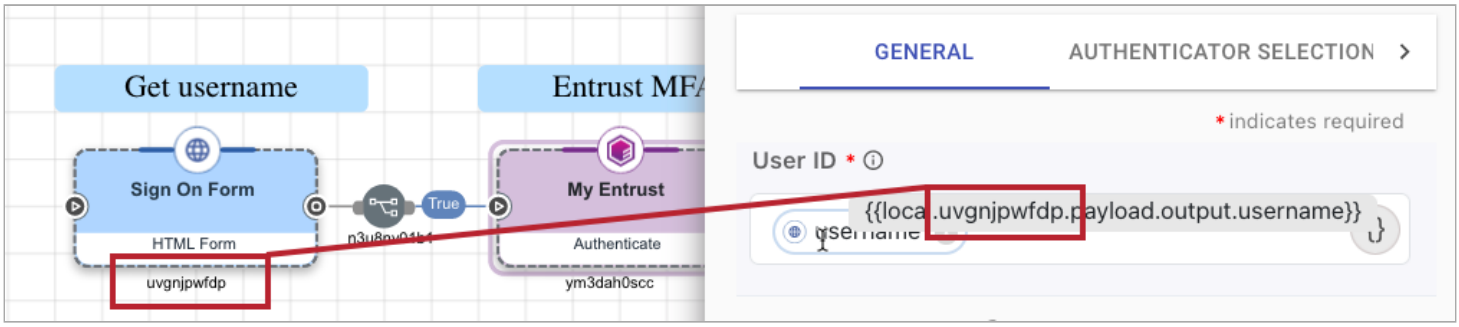
> **Note**
>
> When **Debug Mode** is selected, the logs can include sensitive data, such as credentials or personal information.

Use Node IDs to track nodes

Turn on node IDs to identify nodes in your flow:

1. Open your flow in PingOne DaVinci.
2. In the upper-right corner of the flow editor, click the **More Options (⋮)** icon and click **Show Node ID**.

When you use variables to populate capability properties in your flow, you can hover over the variable to see the ID of the node that the variable comes from. With **Show Node ID** enabled, you can check the source of the variable to make sure it's correct.



Note

Always make sure your variables point to the expected node ID after you copy or clone nodes or flows.

Review API responses

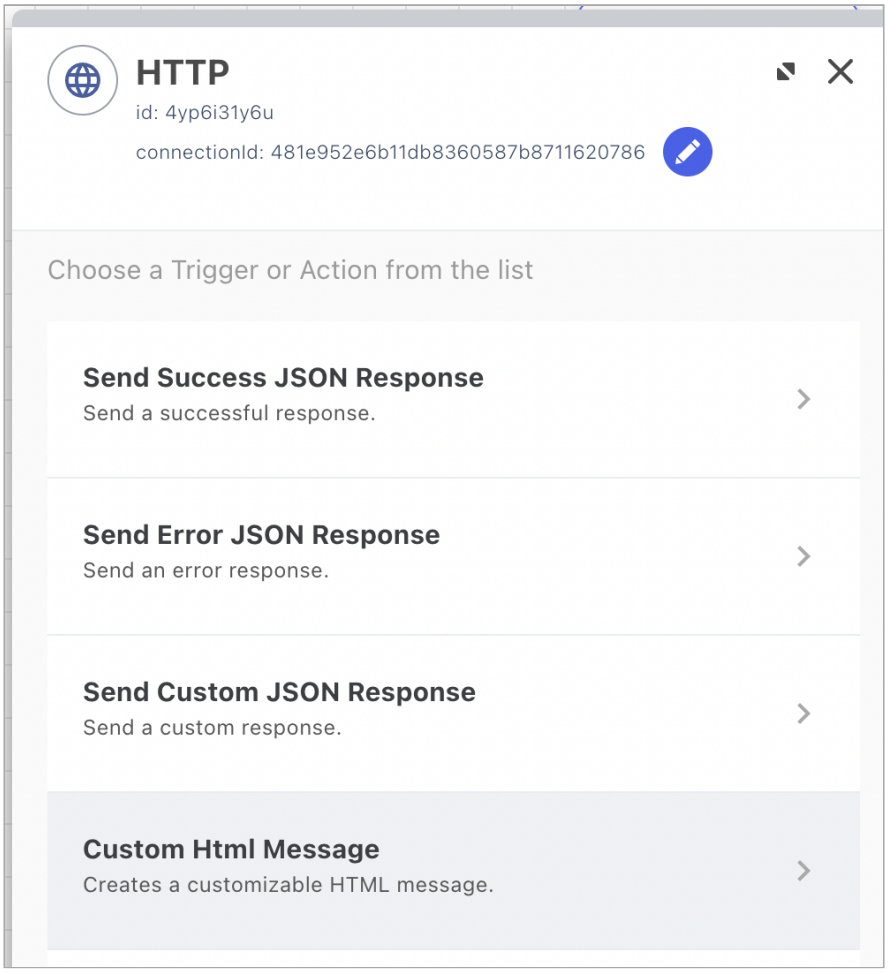
Learning how to read API responses is also useful for troubleshooting in PingOne DaVinci.

If you scroll to the top of an event, you can review the properties of the API call and the schema for the connector.

Use the Error Message connector

The Error Message connector allows you to display custom error messages in a DaVinci flow and is useful for handling errors consistently in a production flow.

For debugging a test flow, use the HTTP connector with the **Custom HTML Message** capability to capture and display errors.

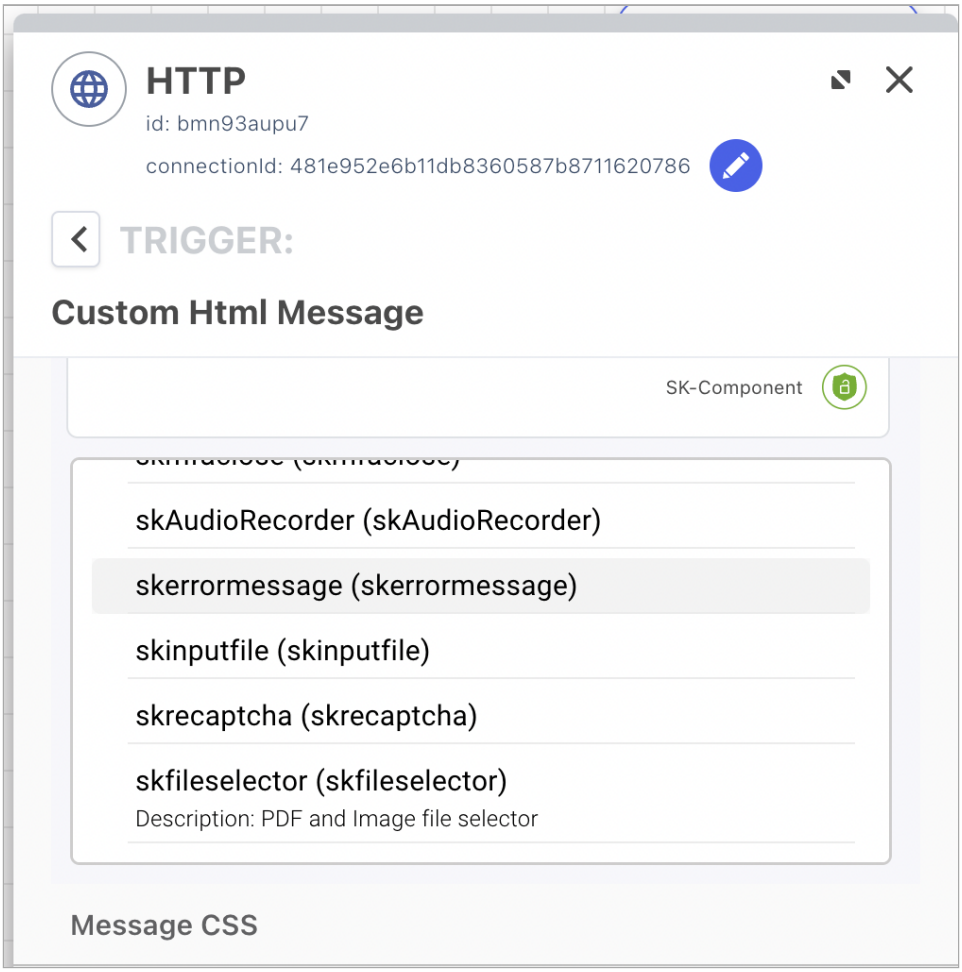


When using the HTTP connector, use the **skerrormessage** SK-component to capture and show DaVinci-specific error information.

There are two ways to display the **skerrormessage**.

Use a basic message:

1. In your flow, add an HTTP connector and select the **Custom HTML Message** capability. Select the node on the canvas.
2. In the **Message** field, click **{}** and select the **skerrormessage** variable from the **SK-Component** source.



Use a custom HTML template:

1. In your flow, add an HTTP connector and select the **Custom HTML Template** capability. Select the node on the canvas.
2. In the **Message** field, click **{}** and select the **skerrormessage** variable from the **SK-Component** source.

For example, if you want to capture the **skerrormessage** on a password validation, HTML similar to the following may be used:

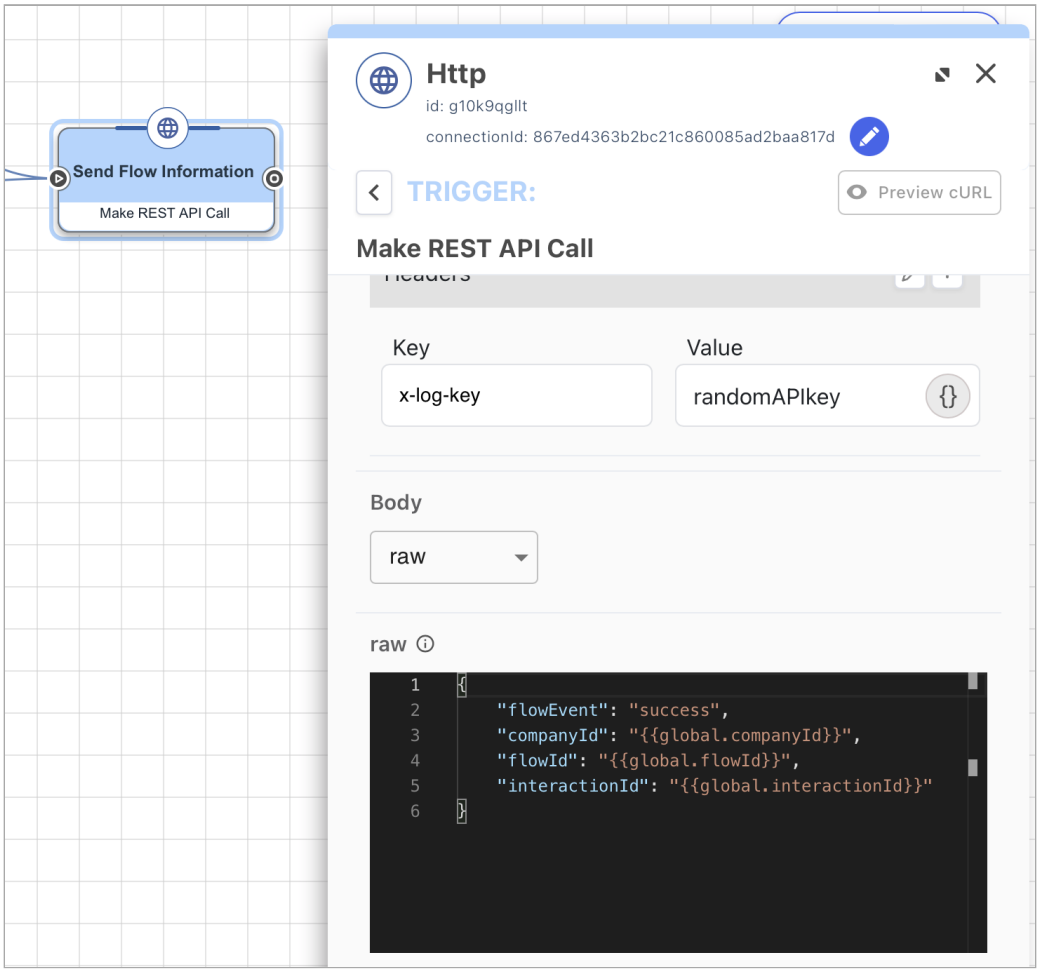
```
<div id="password-validation-message"
  class="errormsg feedback--error sk-alert sk-alert-danger has-text-danger has-background-danger-light text-center"
  style="color:red" data-skcomponent="skerrormessage" data-skerrorid="password">
</div>
<div data-skcomponent="skerror"
  class="feedback feedback--error sk-alert sk-alert-danger has-text-danger has-background-danger-light text-center"
  style="color:red" data-id="feedback" data-skvisibility="">
</div>
```

Send flow data to an external analytics tool

You can use the HTTP connector's **Make REST API Call** capability to send DaVinci log data to an external log aggregator or analytics tool.

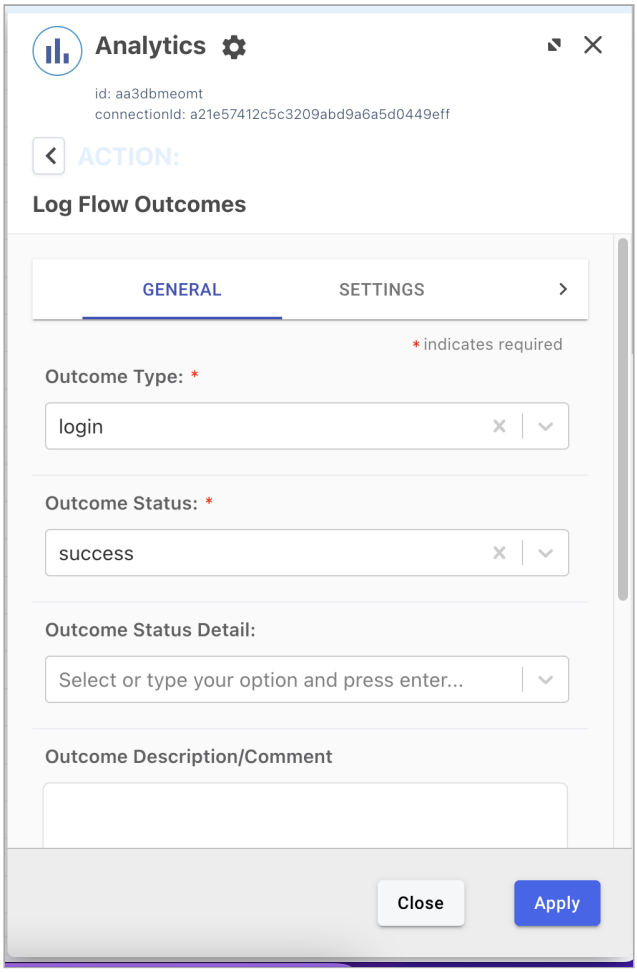
1. In your flow, add an **HTTP connector** and select the **Make REST API Call** capability. Select the node on the canvas.

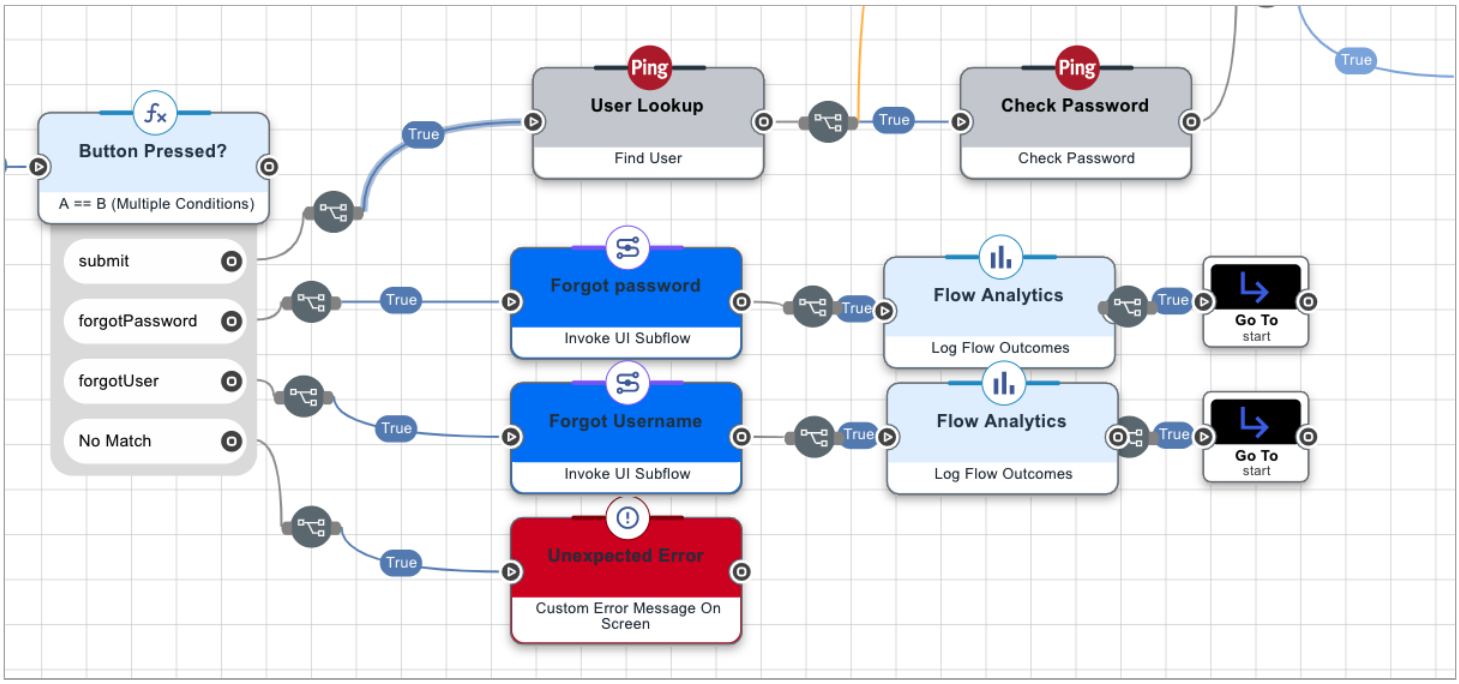
2. In the **Headers** section, click the **Add (+)** icon.
3. In the **Key** field, enter `x-log-key`.
4. In the **Value** field, enter the API key.



Add the Flow Analytics connector to the flow

The Flow Analytics connector enhances the standard analytics view with cumulative results. Use the Flow Analytics connector in key places in your flow to collect the information that's most important to you.





Correlate information using available IDs

Multiple ID values display in the analytics and in webhook events at all logging levels. You can use these IDs to correlate events between products and gain a clearer understanding of issues you encounter.

Identifier	Description	Purpose
`correlationId`	A `correlationId` is generated for each API call. A node that communicates with PingOne uses one or more API calls, such as `/as/authorize`, `/policy/:policyId/start`, or `/capabilities/customHTMLTemplate`.	Lets you trace a specific HTTP request execution within PingOne that spans multiple PingOne microservices.
`transactionId`	A `transactionId` is assigned for each flow execution across all services. For example, if a PingOne flow is initiated from the OAuth authorization endpoint, the `transactionId` would be set at the first step and then passed to DaVinci. Even if the same flow execution returns to PingOne, the same `transactionId` is maintained.	Used to trace a flow execution across all PingOne services.
`externalTransactionId`	A `externalTransactionId` is a unique identifier set outside of PingOne by an external application. It's then passed to PingOne and DaVinci and carried throughout the flow execution. For example, flows executed as part of PingFederate integration are assigned an `externalTransactionId` by PingFederate, and that value is tracked throughout the flow execution.	Used to trace the flow execution across all PingOne and external services when an external system is involved.
`sessionId`	A `sessionId` identifies a unique user session. It's set when the session is created in PingOne.	Used to trace user sessions across flow executions.
`externalSessionId`	An `externalSessionId` identifies a unique user session. It's set by an external application such as PingFederate and passed to PingOne and DaVinci.	Used to trace user sessions across flow executions and external applications.

Change management

Follow these best practices to manage and track changes to your flows.

Use flow aliases for version control

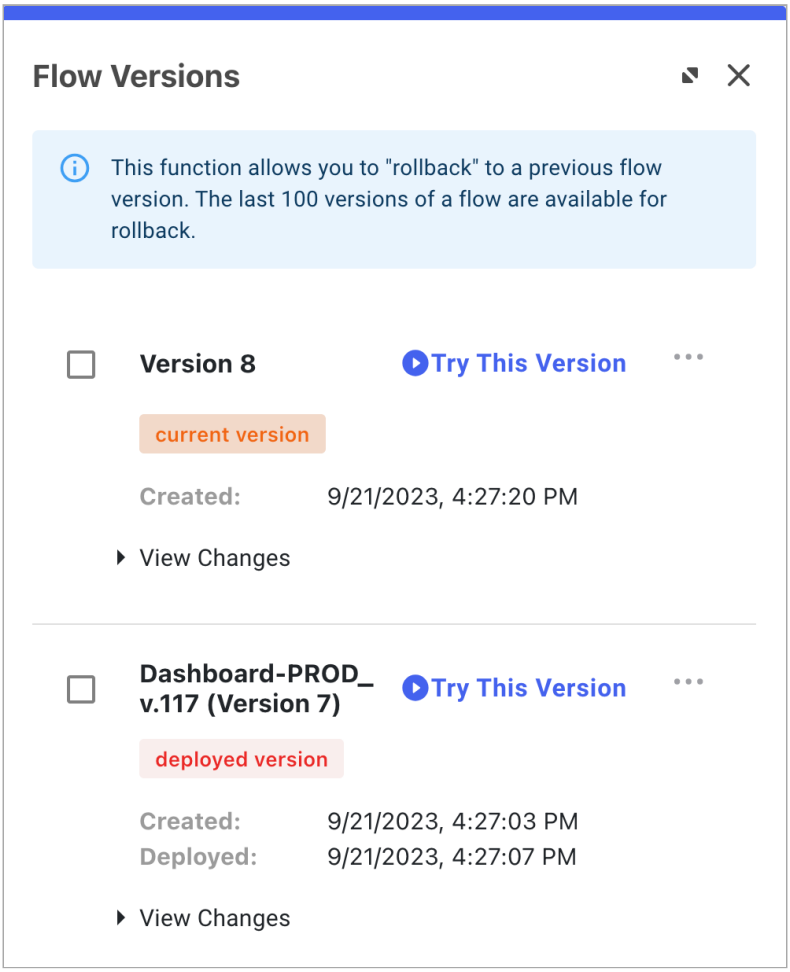
When your flow reaches a milestone, add an alias to the flow version with a name, deployment status, and version number, such as `Dashboard-PROD_v.117`.

1. Open the flow.

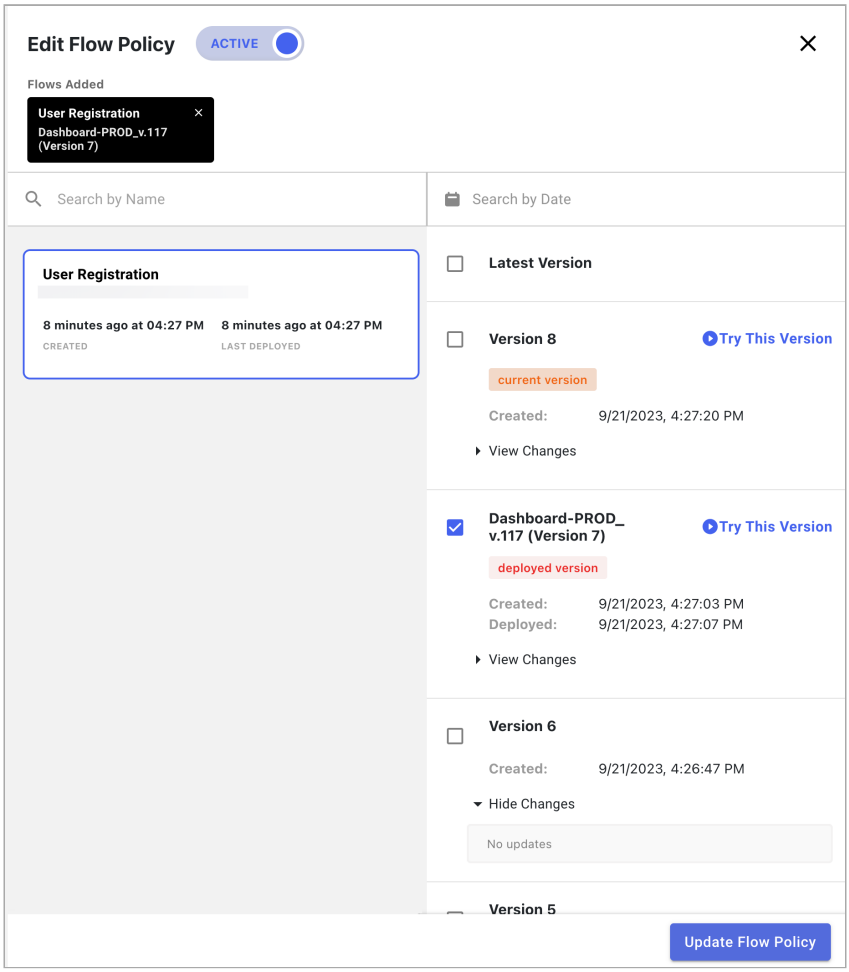
2. Go to **More options (⋮) → Flow Versions**.

3. Find the correct version and go to **... → Set Version Alias**.

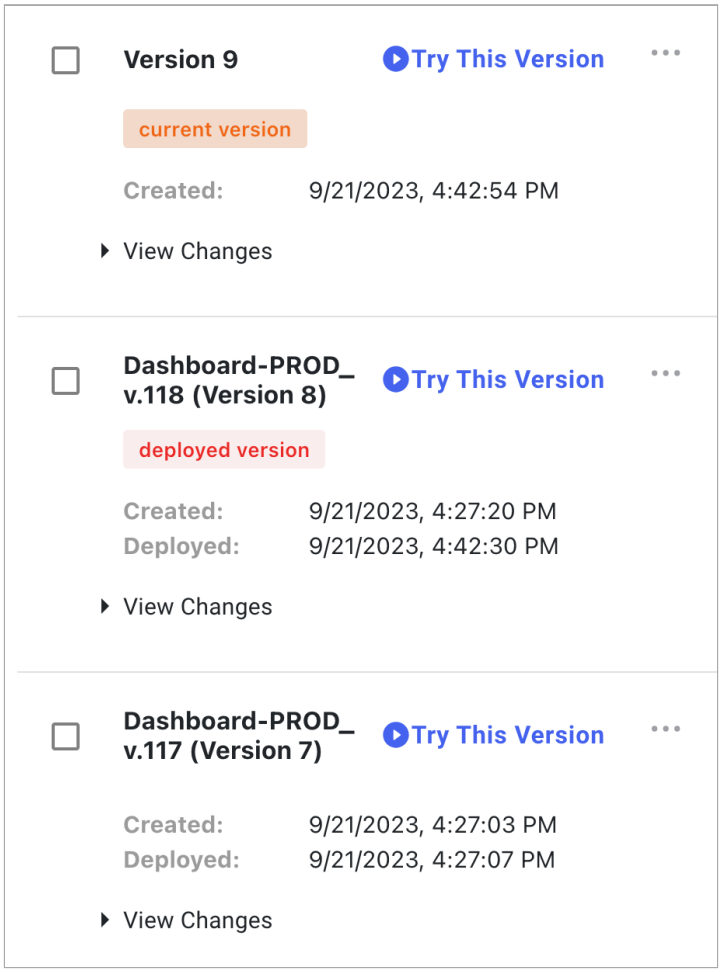
4. Enter the alias and click **Save**.



Configure your application's flow policy to use the specific version of the flow instead of the latest version. This allows you to continue developing your flow without affecting production.



When you change the flow, update the version number in the name, as shown in the following examples.



Export flows and subflows for source control

With each version change of the flow, go to the menu and export the flow as a JSON file:

1. In the upper-right corner of the flow editor, click the **More Options (⋮)** icon and click **Download Flow JSON**.
2. Save the file.
3. Follow your organization's source control best practices for maintaining the file.

> **Note**
>
> Do not post flows or subflows to externally-accessible locations to protect secure information.

Using custom code safely

Follow these best practices to use custom code in your flows.

Custom code is a feature that lets you create your own code to include in DaVinci flows. It's available in multiple connectors and capabilities, including:

- Custom HTML template **script** field

- Custom Functions
- Code Snippet fields

Risks

Because custom code fields can run any code provided, they carry additional security risks.

Learn more about the risks of custom code in the Open Worldwide Application Security Project resources.

Code Execution Location

When you include custom code, it either runs on the server side or the client side, depending on the node type:

- The **Code Snippet** connector and the **Custom Function** capability of the **Functions** connector run code on the server side.
- Script fields in any other node with a customizable HTML template are run on the client side.

You need to consider where the code runs when designing and building flows. Because client-side code can be viewed, you shouldn't include sensitive values in code that is run on the client side.

In addition, variable and parameter values included in custom code have their values added in different ways, which can impact your flow design.

- Variable values included using the `{{global.variables.variableName}}` structure have the value substituted on the server side before the code runs, replacing the structure with the unescaped value or, if the variable value is a object, a stringified version. This means that, if you want to include non-static HTML in your custom code, you should use variables and not parameters.
- Parameters included using the `{{parameterName}}` structure are sent as an additional argument to the client, where Handlebars uses it to process the template. This does escape any special characters in the parameter value.

If you want to use an object or a portion of an object in custom code, you can use the following methods, ordered from most to least secure:

1. Add the input properties to the node's input schema and map the values from the source object.
2. If the number of inputs can't be known ahead of time, create an input property to pass the object, then use Handlebars #each, #with, and lookup helpers to access the individual values.
3. Use an alternate syntax for the object property you want to reference. Replace the periods in the object name with slashes. For example, `{{myUser/category/example}}`.
4. If you want to use JavaScript instead of Handlebars to add content, you can use input parameters using this format: `(const myObject = JSON.parse( {{myInput}} );)` or use DaVinci parameter substitution using this format: `(const myObject = JSON.parse( {{local.aNodeId.aCapability.output.someObject}} );)`.

Recommendations

Follow these recommendations when using custom code fields:

- Make sure that only trusted users can access DaVinci to add custom code.

- Make sure that any custom code you plan to use is reviewed before it is added to user-facing flows, regardless of whether the custom code was produced by you or by a third party. You can use Semgrep as a tool for reviewing your code for security vulnerabilities.
- Make sure that sensitive or private information isn't exposed by any variables or run-time data processed by custom code.

Best practices for gathering support information

Use these best practices to gather information for working with Ping support to resolve issues.

Gather environment information

Find the environment and organization details in PingOne:

1. In the PingOne admin console, go to **Settings > Environment Properties**.
2. Copy the **Environment ID** for later reference.
3. Copy the **Organization ID** for later reference.

Gather flow-specific information

If your issue involves a specific flow, gather information about the flow:

1. Export the flow as JSON:
    1. In DaVinci, go to **Flows**.
    2. Click the flow you want to export.
    3. Click ⋮ > **Download Flow JSON**.
    4. Clear **Include Variable Values**.
    5. Click **Export**.
2. Capture an HTTP trace of the flow, following the HTTP tracing best practices.
3. If the flow is a sample flow created by Ping, note any customization you applied to the flow.
4. If the flow is custom, describe the use case, the intended method of operation, and, if possible, the section or sections causing problems.
5. Update the logging settings to gather additional information:
    1. In DaVinci, go to **Flows**.
    2. Click the flow, then click ⋮ > **Flow Settings**.
    3. Click the **Logging** tab.
    4. Set the **Log Level** to **Debug**.
    5. If the flow normally contains sensitive information, select **Scrub Sensitive Information**.

6. If you can identify specific flow executions or nodes where the problem occurs, gather information about the specific error:
    1. In DaVinci, go to **Flows**.
    2. Click the flow.
    3. Click **Analytics**.
    4. In the timeframe selector, select a timeframe that includes the relevant executions.
    5. In the **Events Logs** section, select a search parameter and locate a flow execution affected by the error.
    6. Note the **Flow Execution ID**.
    7. If possible, note the **User ID** or **User Name** of the user associated with the flow execution.
    8. If you can identify the node or nodes causing the error, take a screenshot of the node. Locate the response event for the node and note any additional details such as error codes or messages. If it's a PingOne connector, note the **Correlation ID** for the node execution.

Integrating Flows into Applications

After you create a flow, integrate it into a user-facing application. Integrating a flow into an application lets your users launch the flow from that application.

You can integrate a flow in different ways. Each method launches the flow in a different way. Choose an integration method based on the type of flow that you want to launch and the desired user experience.

The following methods can be used to launch a flow:

- A redirect through PingOne. This method uses a call to a PingOne application to launch a flow with a redirect. This method is effective for flows with UI components. You should use a redirect through PingOne if you want to launch the flow in a new application page that replaces the current page and if you want to use OpenID Connect (OIDC) or Security Assertion Markup Language (SAML) authentication.
- A redirect through PingOne using DaVinci as an external identity provider (IdP). This method uses a call to a PingOne application to launch a flow with a redirect using an external IdP configuration. This method is effective for flows with UI components, but it's not recommended unless you have already configured your environment for it. If you want to configure your environment to launch flows with a redirect through PingOne, use this procedure instead.
- The widget. This method launches a flow inside of a widget on the current page. This method is effective for situations in which you do not want to redirect the user to a new URL.
- An API call. This method launches a flow using an API call. This method is effective for flows without a UI component.
- The SDK. This method launches a flow from an application that you develop using the DaVinci module for the Ping SDK for JavaScript, Ping SDK for iOS, or Ping SDK for Android. This method is appropriate if you want fine-grained control of a user's mobile experience.
- The PingFederate integration. This method uses the widget to launch a flow from an existing PingFederate deployment.

> **Note**
>
> To switch between using flows for a PingOne redirect integration and an integration using the DaVinci widget, see Switching between PingOne and DaVinci widget integrations.

Integration method comparison

	Redirect	Widget	API	SDK	DaVinci Integration Kit for PingFederate (widget mode)	DaVinci Integration Kit for PingFederate (API mode)
Description	Launches flow in new browser tab	Launches flow within current browser tab	Launches flow without UI components using API call	Launches flow from native or web apps using the Ping SDKs from a PingOne application	Launches flow within current browser tab	Launches flow without UI components using API call

	Redirect	Widget	API	SDK	DaVinci Integration Kit for PingFederate (widget mode)	DaVinci Integration Kit for PingFederate (API mode)
Trigger	PingOne Policy link	Widget embedded in application	API Call	Launch from within SDK application	PingFederate authentication policy initiates DaVinci Integration Kit adapter in widget-based flow mode	PingFederate authentication policy initiates DaVinci Integration Kit adapter in API-based flow mode
UX hosted by	DaVinci	Application that launched the flow	None Launching application must handle UX	Custom application built with the SDK	PingFederate	None DaVinci adapter in API-based flow mode does not present a UI
HTML	DaVinci using PingOne Forms or Custom HTML	DaVinci with Custom HTML		Launching application must handle UX	DaVinci Integration Kit template	
CSS	DaVinci	Host application		Launching application must handle UX	DaVinci Integration Kit template	
User Experience	User's browser tab is redirected to DaVinci with refresh	Flow is launched within the host application		Flow is launched within the application	Flow is launched within PingFederate	
Modes	Full screen	Embedded in host application or modal		Depends on launching application	Embedded in DaVinci Integration Kit template	

	Redirect	Widget	API	SDK	DaVinci Integration Kit for PingFederate (widget mode)	DaVinci Integration Kit for PingFederate (API mode)
Developer experience for launching flows **Note** This applies only to launching flows. Flow creation is equally complex for all methods.	No development skills needed Flow hosted in DaVinci	Minimal development effort Flow is a component in existing application	Significant development effort	Significant development effort	Minimal development effort • Template presents flow • Customizations optional	No development skills needed Flow result data is available in PingFederate authentication policy
Time required	Fastest	Fast	Slow	Slow	Fast (with existing PingFederate deployment)	Fastest (with existing PingFederate deployment)

Launching a PingOne flow with a redirect

You can configure PingOne and DaVinci so that you can invoke specifically-configured flows through PingOne.

This approach lets you launch your flows from PingOne and lets you reference and modify user data from PingOne within the flow.

> **Note**
>
> To switch between using a flow for a DaVinci widget integration and an integration using a PingOne redirect, see Switching between PingOne and DaVinci widget integrations.

Configuring a DaVinci flow for invocation

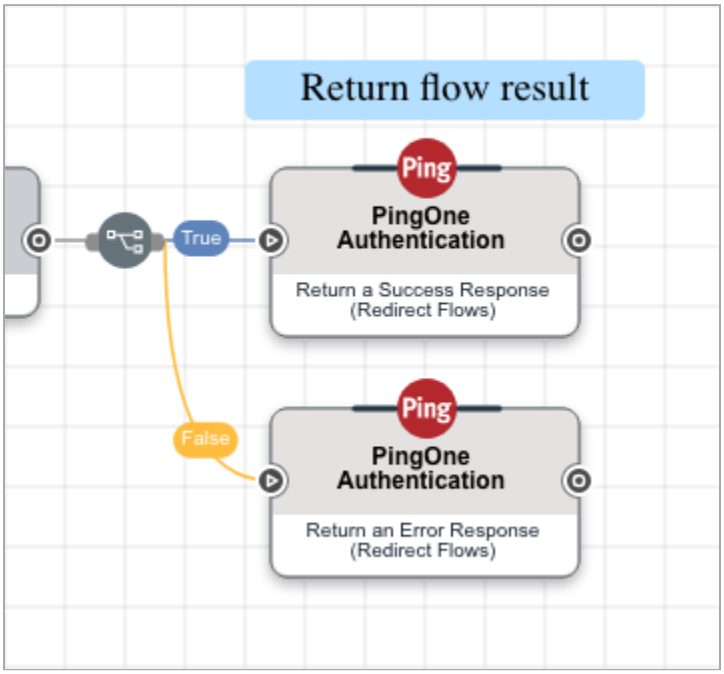
Update a DaVinci flow to enable it to be launched through PingOne.

About this task

This procedure assumes that the flow already exists or is in progress, and only specifies the necessary nodes and settings for invocation through PingOne. For more information about creating flows, see Getting started with DaVinci.

Steps

1. Sign on to DaVinci and click the **Flows** tab.
2. Select the flow tile for the flow that you plan to launch through PingOne.
3. Click **More options (⋮) → Flow Settings** to show the flow settings.
4. Select the **PingOne Flow** option.
5. Click **Save**, then close the **Flow Settings** pane.
6. End the flow with the following two **PingOne Authentication** nodes, one for success and one for failure.



Node	Purpose
Return a Success Response (Redirect Flows)	This creates a PingOne session for the user and redirects the browser back to the source of the authentication request. This response provides the requested scopes as well as an access token, ID token, or SAML assertion.
Return an Error Response (Redirect Flows)	This redirects the browser back to the source of the authentication request. This response provides information about the error that occurred.

7. Click **Save**, then click **Deploy**.

Using PingOne connectors

Add PingOne connectors to your environment to use PingOne capabilities in your flows.

For information about the available PingOne connectors, see Core connectors. If you plan to launch flows through PingOne, you must add the PingOne Authentication connector to your environment.

Referencing PingOne data in the flow

You can reference data from PingOne within your flow.

The format for this information is `global.parameters.<parameter name>`. Some parameters depend on the protocol used to launch the flow, while others are supplied by the PingOne application.

OIDC parameters

Property	Description
`authorizationRequest`	An object that specifies all the parameters from the OIDC authorization request.
`authorizationRequest.<customParameter>`	A string that specifies a custom URL parameter added to the OIDC authorization request. Replace *<customParameter>* with the name of the custom URL parameter.
`authorizationRequest.client_id`	A string that specifies the client ID of the application associated with this authorize request.
`authorizationRequest.redirect_uri`	A string that specifies the URL of the return entry point of the application.
`authorizationRequest.response_type`	A string that specifies the code or token type returned by an authorization request. Options are `token`, `id_token`, and `code`.
`authorizationRequest.scope`	A string that specifies the permissions that determine the resources that the application can access.
`authorizationRequest.state`	A string that maintains the state between the logout request and the callback to the endpoint specified by the `post_logout_redirect_uri` query parameter.
`authorizationRequest.nonce`	A string that is used to associate a client session with a token to mitigate replay attacks. The value is passed through unmodified from the authentication request to the token. This is an optional property for authorization requests that return a code.
`authorizationRequest.acr_values`	A string that is used by the flow designer to pass in useful information.
`authorizationRequest.login_hint`	A string that is used to designate a login identifier to pre-fill the **username** field of the sign-on screen.
`authorizationRequest.max_age`	A string that specifies the maximum amount of time allowed (in seconds) since the user last authenticated. If the `max_age` value is exceeded, the user must re-authenticate. If the `max_age` value is set to 0 (`max_age=0`), the user is always required to re-authenticate.

Property	Description
`authorizationRequest.prompt`	A string that specifies whether the user is prompted to sign on for re-authentication. The prompt parameter can be used as a way to check for existing authentication, verifying that the user is still present for the current session.
`authorizationRequest.response_mode`	A string that specifies the mechanism for returning authorization response parameters from the authorization endpoint. Options are `query`, `fragment`, and `form_post`.
`authorizationRequest.code_challenge`	A string that is computed from the `code_verifier` that is used in a Proof Key for Code Exchange (PKCE) authorization request.
`authorizationRequest.code_challenge_method`	A string that specifies the computation logic used to generate the `code_challenge` string. The token endpoint uses this method to verify the `code_verifier` for PKCE authorization requests. Options are `plain` and `S256`.
`authorizationRequest.code_verifier`	A string that is used to create the `code_challenge` value passed to the authorization server in the request.
`authorizationRequest.mobileRequest`	An object that specifies OIDC/OAuth2 request parameters.

For example, the following code references the login hint in a flow launched using OIDC:

```
global.parameters.authorizationRequest.login_hint
```

SAML parameters

Property	Description
`samlRequest`	A string that specifies all the parameters from the SAML request.
`samlRequest.spEntityId`	A string that specifies the service provider entity ID used to look up the application. This is a required property and is unique within the environment.
`samlRequest.forceAuthn`	A boolean that, when set to true, specifies that the identity provider must authenticate the presenter directly rather than rely on a previous security context. If a value is not provided, the default value is false.
`samlRequest.passive`	A boolean that, when set to true, specifies that the identity provider and the user agent itself must not visibly take control of the user interface from the requester and interact with the presenter in a noticeable fashion. If a value is not provided, the default value is false.

Property	Description
`samlRequest.signed`	A boolean that specifies whether the SAML assertion should be signed. The default value is false.
`samlRequest.subject`	A string that specifies the SAML subject ID.
`samlRequest.requestedAuthnContext`	A string that specifies the authentication methods for the request.

For example, the following code references the subject ID in a flow launched using SAML:

```
global.parameters.samlRequest.subject
```

WS-Federation parameters

Property	Description
`wsFedRequest.wfresh`	The maximum age of authentication in minutes. If the value is 0, the user should be prompted for authentication before a token is issued.
`wsFedRequest.wauth`	The required authentication level.
`wsFedRequest.wctx`	An opaque context value that can be passed in the request. When the invoked DaVinci flow completes successfully and returns to PingOne, PingOne echoes this value back to the service provider with the issued token if it was included by the service provider in the originating request.
`wsFedRequest.whr`	The account partner realm of the client.

Note

The WS-Federation parameters are currently in limited release. To request access to these parameters, open a support case.

Application parameters

Property	Description
`application`	An object that specifies the configuration information about the PingOne application that initiated the authentication request.

Property	Description
`application.homePageUrl`	A string that specifies the custom home page URL for the application.
`application.id`	A string that specifies the application ID.
`application.name`	A string that specifies the application name.
`application.protocol`	A string that specifies the protocol for the application. Options are OPENID_CONNECT and SAML.
`application.type`	A string that specifies the application type. Valid values are `WEB_APP`, `NATIVE_APP`, `SINGLE_PAGE_APP`, `SERVICE`, `CUSTOM_APP`, `WORKER`, `PING_ONE_SELF_SERVICE`, `PING_ONE_ADMIN_CONSOLE`, `PING_ONE_PORTAL`, `TEMPLATE_APP`, and `PORTAL_LINK_APP`.

For example, the following code references the application ID in a flow, regardless of the protocol used to launch the flow:

```
global.parameters.application.id
```

Universal parameters

Property	Description
`loginHint`	A string that specifies an identifier to pre-fill the **username** field of a sign-on screen.
`maxSecondsSinceLastSignOn`	An integer that specifies the maximum amount of time allowed (in seconds) since the user last authenticated. If the user's last sign on in the session is greater than the integer value specified in this property, then existing session information cannot be used to skip authentication or influence any authentication logic. This value is set automatically to 0 if `prompt=login` is set for an OIDC application or if `ForceAuthn=true` is set for a SAML application. Otherwise, this value is set to the `max_age` property value for OIDC applications, if present, or omitted otherwise.

For example, the following code references the login hint, regardless of the protocol used to launch the flow:

```
global.parameters.loginHint
```

Configuring a DaVinci flow policy for invocation

Configure a flow policy to specify which flow and which version of the flow you want to launch.

About this task

Flows in DaVinci flows often have multiple versions as administrators make changes, and not all of these versions should be presented to users. A flow policy ensures that users see the correct version of the correct flow.

Steps

1. Sign on to DaVinci and click the **Applications** tab.
2. Find the application and click **Edit**.
3. Click the **Flow Policy** tab.
4. Click **Add Flow Policy**.
5. In the **Name** field, enter a name for the flow policy.
6. Select **PingOne Flow Policy**.
7. Add one or more PingOne flows to the policy.

> **Important**
>
> PingOne flow policies can only include flows and flow versions that have the **PingOne Flow** setting enabled. Flows and versions without this setting cannot be selected.

    1. In the **Flows** section, select a flow.
    2. In the **Version** section, select one or more versions of the flow.

       The **Latest Version** option always uses the latest version.

    3. **Optional:** Repeat the previous steps to add additional flows.
8. Click **Create Flow Policy**.

   The **Edit Your Weight Distribution** modal opens.
9. Add weight distribution and analytics information for each flow and flow version:
    1. In the **Distribution** field for each flow version, enter or select a distribution weight from 1 to 100.

> **Note**
>
> The weight is used if the flow policy is invoked without a flow ID. You can use the weight to perform A/B testing.

    2. **Optional:** Click **Add IP Whitelist**.
    3. **Optional:** In the **Whitelist IP** field, enter one or more IP addresses.

> **Note**
>
> If a request comes from an allowed IP address, the weight is ignored, and the specified flow is triggered.

4. **Optional:** In the **Analytics - Select Success Nodes** list, select one or more nodes that, when run, indicate that the flow run was successful.

   This information is used to calculate the flow policy's success rate.

10. Click **Save Flow Policy**.

Configuring PingOne for flow invocation

Configure an application in PingOne to launch flows.

About this task

The properties of the PingOne application are used as part of the URL that launches the flow.

Steps

1. Sign on to PingOne and go to **Applications > Applications**.
2. Click the + icon.
3. In the **Application Name** field, enter a name for the application.
4. In the **Application Type** section, select **OIDC Web App** or **SAML Application**.
5. If you selected **SAML Application**, provide the SAML configuration.
   1. Click **Configure**.
   2. Select a method for providing the application metadata.

      Choose from:

      - **Import Metadata**: Import the configuration details from an XML metadata file. Click **Select a File** and then select an XML metadata file on your system. Click **Open**.

        If the metadata file does not specify all the configuration values, you must enter the missing values manually.

      - **Import From URL**: Import the configuration details from a metadata URL. Enter the URL and then click **Import**.

        The URL must be a valid absolute URL.

      - **Manually Enter**: Enter the configuration details manually. In the **ACS URLs** field, enter the Assertion Consumer Service (ACS) URLs. You must specify at least one URL, and the first URL in the list is used as the default. In the **Entity ID** field, enter the service provider entity ID used to look up the application.
6. Click **Save**.
7. Click the **Policies** tab.
8. Click + **Add Policies**.

9. Click the **DaVinci Policies** tab.
10. Select one or more flow policies to add to the application.

    Only flow policies with the **PingOne Policy** option are displayed.
11. Click **Save**.
12. Click the **Configuration** tab.
13. Click the **Pencil** icon.
14. In the **CORS Settings** section, select **Allow specific origins**.
15. In the **Allowed Origins** field, enter the domain from which you plan to launch the flow.
16. Click **Save**.

Invoking the flow

To launch the flow, construct a link with the PingOne details and add it to the resource that will launch the flow.

Steps

1. In the resource that will launch the flow, such as your organization's web application, add a call to your PingOne application using the following format:

```
https://auth.pingone.com/<Environment ID>/as/authorize?response_type=<response type>
     &client_id=<client ID>
     &redirect_uri=<redirect URI>
     &scope=<scope>
     &acr_values=<Flow Policy ID>
     &<other parameter>=<value>
```

   The following parameters are used in the call:

Table 1. Parameters

Parameter	Required	Description	Location
Environment ID	Yes	The **Company ID** of the DaVinci application.	Available in DaVinci in the **Company** tab, or in the details section at the top of any flow or application.
Response Type	Yes	The response type expected by the PingOne application.	Available in PingOne under **Applications → Applications**. Click your application, then click the **Configuration** tab and find the **Response Type** field.

Parameter	Required	Description	Location
Client ID	Yes	The PingOne application's Client ID.	Available in PingOne under **Applications → Applications**. Click your application, then click the **Configuration** tab and find the **Client ID** field.
Redirect URI	Yes	A redirect URI configured in PingOne.	Available in PingOne under **Applications → Applications**. Click your application, then click the **Configuration** tab and find the **Redirect URIs** field.
Scope	Yes	The application request scope.	Available in PingOne under **Applications → Applications → Resources**. Click your application, then click the **Resources** tab and find the **Allowed Scopes** section.
Flow Policy ID	No	A policy that determines which flow and version is run.	Available in DaVinci in the **Applications** tab. Select your application, then click the **Flow Policy** tab.
Additional parameters	No	You can pass in additional parameters to make their values available during the flow.	N/A

Note

You can reference the parameter values passed in with the invocation. The format is:

```
global.parameters.authorizationRequest.<parameter name>
```

2. If the user requires a token but the flow did not grant a token, make an API call to the PingOne token endpoint to grant the user a token. Use the PingOne token authentication code endpoint or the Pingone client credentials endpoint.

Launching a PingOne flow with a redirect using an external IdP

You can configure PingOne and DaVinci so that you can invoke specifically-configured flows through PingOne. This method configures DaVinci as an identity provider (IdP) in PingOne.

Note

If you have not yet configured PingOne to launch DaVinci flows, follow the steps in Launching a PingOne flow with a redirect instead.

For additional information about integrating PingOne with DaVinci, see the DaVinci Application Integrations guide.

Creating PingOne credentials

Create a set of credentials for adding PingOne connections.

Steps

1. Sign on to PingOne.
2. Create a worker app as described in the PingOne documentation.
3. Assign the following roles to the worker app:
    - Identity Data Admin
    - Environment Admin
4. Note the **Client ID**, **Client Secret**, and **Environment ID** for the worker app.
5. Click **Finish**.
6. Go to **Applications → Applications**, click the application to open the application details, and click the toggle switch in the upper right to enable the application.

Adding PingOne connectors

Add one or more PingOne connectors in DaVinci to enable your flows to use PingOne-related capabilities such as reading or updating users.

Steps

1. Sign on to DaVinci.
2. Click the **Connectors** tab.
3. Click **Add Connector**.
4. In the list of available connectors, select one of the PingOne connectors and click **+**.
    - PingOne
    - PingOne Authentication
    - PingOne Forms
    - PingOne Notifications
    - PingOne RADIUS Gateway
    - PingOne Authorize
    - {davinci-pingone-credentials-connector}
    - {davinci-pingone-mfa-connector}
    - {davinci-pingone-risk-connector}

- {davinci-pingone-verify-connector}

The **New Connector** modal opens.

5. Enter a name for the new connector and click **Create**.
6. Find and click the newly-created connector in the list of your connectors.
7. Set up the connector configuration.

> **Note**
>
> The **Client ID**, **Client Secret**, and **Environment ID** that you noted in the previous procedure are used to configure the PingOne connectors.

8. Click **Apply**.

Preparing a flow

Create a flow, then prepare it for implementation through an OpenID Connect (OIDC) call to PingOne.

> **Note**
>
> This procedure does not go into detail about creating a flow. See the other use cases for additional information about creating specific flows. The preparation steps apply regardless of the purpose of your flow.

This implementation method launches the flow in a new page. The user is redirected to the flow, which replaces the previous page and uses a DaVinci URL.

This implementation method is simple and does not require the addition of a widget to the page. It's also well-suited for any type of flow. Because the flow takes the entire page, you must design the flow to match your own branding and style.

Depending on the purpose of the flow, you might want to include a token connector, which directs PingOne to mint a token for the user.

You can reference information from PingOne in your flow. The format for this information is `global.skOpenId.p1Oidc.<request object hierarchy>`, where the hierarchy is taken from the request object schema.

The full request object schema for OIDC is:

```
"p1Oidc": {
  "id": "<ID value>",
  "environment": {
    "id": "<environment ID value>"
  },
  "application": {
    "id": "<application ID value>"
  },
  "user": {
    "id": "<user ID value>"
  },
  "request": {
    "http": {
      "remoteIp": "<remote IP value>",
      "userAgent": "<user agent information>",
      "headers": {
        "Accept-Language": [
          "<language value>"
        ]
      }
    },
    "oidc": {
      "responseTypes": [
        "<response type>"
      ],
      "acrValues": [
        "<acr value>"
      ],
      "scopes": [
        "<scope value>"
      ],
      "parameters": {
        <One or more parameter-value pairs>
```

The full request object schema for Security Assertion Markup Language (SAML) is:

```
"p1Oidc": {
    "id": "<ID value>",
    "environment": {
        "id": "<environment ID value>"
    },
    "application": {
        "id": "<application ID value>"
    },
    "request": {
        "http": {
            "remoteIp": "<remote IP value>",
            "userAgent": "<user agent information>"
        },
        "saml": {
            "environmentId": "<environment ID value>",
            "urlContext": {
                "environmentId": "<environment ID value>",
                "customDomainHost": "<custom domain host value>,
                "hostUrl": "<host URL value>",
                "authHostUrl": "<URL value>",
                "idpEntityId": "<IDP entity ID value>",
                "internalIdpEntityId": "<internal IDP entity ID value>",
                "flowHeaders": {
                    "X-Forwarded-Host": "<host value>"
                },
                "host": "<host name>",
                "customDomain": <domain value>
            },
            "ssoInitialized": <value>,
            "requestBinding": <value>,
            "requestSigned": <value>,
            "ssoRequest": {
                "requestId": <ID value>,
                "version": <value>,
                "issueInstant": <value>,
                "destination": <value>,
                "consent": <value>,
                "spEntityId": "<security ID value>",
                "forceAuthn": <value>,
                "passive": <value>,
                "acsUrl": "<value>",
                "acsBinding": "<value>",
                "subject": <value>,
                "signed": <value>,
                "authnContextRef": <value>
            },
            "application": {
                "id": "<ID value>",
                "name": "<name value>",
                "protocol": "<value>",
                "enabled": <value>,
                "spEntityId": "<value>",
                "acsBinding": "<value>",
                "acsUrls": ["<value>"],
                "assertionDuration": <value>,
                "sloBinding": "<value>",
                "assertionSigned": <value>,
                "responseSigned": <value>,
                "environment": {
                    "id": "<value>"
```

```
                }
            },
            "user": <user name value>,
            "relayState": <value>,
            "idpIssuer": "<issuer value>",
            "attributes": <value>
        }
    }
}
```

For example, the following code references the remote IP:

```
global.skOpenId.p1Oidc.request.http.remoteIp
```

> **Note**
>
> Any property you reference must be included by the configured scopes, as described in the Configuring an external IDP section.

Using PingOne connectors

Add PingOne connectors to your environment to use PingOne capabilities in your flows.

For information about the available PingOne connectors, see Core connectors. If you plan to launch flows through PingOne, you must add the PingOne Authentication connector to your environment.

Creating an application

Create an application in DaVinci to enable your flow.

Steps

1. Sign on to DaVinci.
2. Click the **Applications** tab.
3. Click **Add Application**.

   The **Add Application** modal opens.
4. In the **Name** field, enter a name for the application.
5. Click **Create**.
6. Find the application and click **Edit**.
7. On the OIDC tab, note the application parameters for the following:
   - **Company ID**
   - **Client ID**
   - **Client Secret**

- **Issuer**
- **Token Endpoint**
- **JWKS Endpoint**

8. Create a flow policy:
    1. Click the **Flow Policy** tab.
    2. Click **+ Add Flow Policy**.
    3. In the **Name** field, enter a name for the flow policy.
    4. In the flow list, select your flow.
    5. In the version list, select **Latest Version**.
    6. Click **Create Flow Policy**.

       The **Edit Your Weight Distribution** modal opens.

       > **Note**
       >
       > This example only uses one flow, but if your flow policy included multiple flows or flow versions, you could use this modal to split traffic between them.

    7. Click **Save Flow Policy**.
    8. Note the **Policy ID** of your flow policy.

Configuring an external IDP

Configure DaVinci as an external IdP in PingOne.

Steps

1. Sign on to PingOne.
2. Add DaVinci as an OIDC identity provider according to the PingOne documentation.
    1. For the **Connection Details**, use the values that you noted in Creating an application:
        - **Client ID**
        - **Client Secret**
    2. In the **Discovery Document URI** field, enter the well-known endpoint to configure the discovery detail values. The format is:

       ```
       https://auth.pingone.com/<EnvironmentID>/davinci/.well-known/openid-configuration
       ```

    3. In the **Requested Scopes** field, add a scope for each entity you want to import from the PingOne flow. The format for these scopes is:

`p1FlowRequest:<parent entity>.<entity>`

The entity name and parent entities are determined by the object request schema documented above. For example, to make the remote IP available, add the scope `p1FlowRequest:http.remoteIp`.

You can add the `p1FlowRequest` scope to make all entities from the PingOne flow available, but this can sometimes result in request size errors.

> **Note**
>
> Do not configure the **User Information Endpoint**.

3. Create the authorization endpoint using the following structure:

```
https://<domain>/<companyID>/davinci/policy/<policyID>/authorize
```

Use the values that you noted in the previous procedure:

- **Company ID**
- **Policy ID**

4. Create the external IdP sign-on policy step according to the PingOne documentation.
    1. In the **External Identity Provider** list, select the external IdP application you created in step 1.
    2. In the **Required Authentication Level** field, enter `policyId-<your policy ID>`. For example, `policyId-69b043b9edeb60b6c1945617ab1b4fae`.
    3. Select **Pass user context to provider**.
    4. Select the external IdP application, and then click **Save** to save your changes.
5. Create an application in PingOne and assign the sign-on policy step to that application according to the PingOne documentation.
6. Add the referring domain to your new application.
    1. Go to **Applications** > **Applications**.
    2. Select your new application.
    3. Click the **Configuration** tab.
    4. Click the **Pencil** icon.
    5. In the **CORS Settings** section, select **Allow specific origins**.
    6. In the **Allowed Origins** field, enter the domain from which you plan to launch the flow.
    7. Click **Save**.
7. Copy the **Callback URL** for the external IdP in PingOne.

8. (Optional) Copy the JWKS information to enable PingOne context information to be used by DaVinci.
    1. Copy the PingOne Application **JWKS URL**.
    2. Access the JWKS URL and copy the complete JWKS key.
9. Sign on to DaVinci.
10. Click the **Applications** tab.
11. Find the application that you previously created and click **Edit**.
12. Click the **OIDC** tab, and then add the copied callback URL value to the **Redirect URLs** field.
13. **Optional:** Enter the JWKS information to enable DaVinci to use context information from PingOne within flows.
    1. Click **Applications**.
    2. Open your application.
    3. Click the **OIDC** tab.
    4. In the **Service Provider (SP) JWKS URL** field, enter the JWKS URL.
    5. In the **Service Provider (SP) JWKS Keys to Verify Authorization Request Signature** field, enter the JWKS key.

Invoking the flow

Add a link to the resource that invokes the flow using a call to PingOne.

Steps

1. Open the source file for the resource that will launch the flow.
2. Create a call to your PingOne application according to the PingOne documentation and add it to the launching resource.

   The general format used for this call is:

   ```
   https://auth.pingone.com/<Environment ID>/as/authorize?response_type=<response type>
        &client_id=<client ID>
        &redirect_uri=<redirect URI>
        &scope=<scope>
   ```

   > **Note**
   >
   > You can reference the parameter values passed in with the invocation. The format is:
   >
   > ```
   > global.skOpenId.<parameter name>
   > ```

3. If the user requires a token but the flow did not grant a token, make an API call to the PingOne token endpoint to grant the user a token. Use the PingOne token authentication code endpoint or the Pingone client credentials endpoint.

Launching a flow with the widget

Launch a prepared flow with a widget.

This method is effective when you want to launch the flow within the current page instead of redirecting the user.

> **Note**
>
> - You can't launch the DaVinci widget in an iframe.
> - You can find information on switching between using a flow for a PingOne redirect integration and an integration using the DaVinci widget in Switching between PingOne and DaVinci widget integrations.

Configuring the Flow

Prepare a flow to be launched with the widget.

About this task

> **Note**
>
> - This procedure only covers the steps and nodes required to prepare a flow for widget invocation. It assumes that you have already created a flow for the purpose you have in mind. See the Getting started with DaVinci and DaVinci Best Practices documentation for more information about building flows.

Steps

1. Click the **Flows** tab.
2. Find the flow and click **... → Edit**
3. Add a nonce to the input schema.

   > **Note**
   >
   > When you launch the flow, you provide a nonce value. This value is returned when the flow completes, letting you verify the flow's completion.

   1. Click **Flow Options (⋮) → Input Schema**.
   2. Click **Add**.
   3. In the **Parameter Name** field, enter `nonce`.
   4. In the **Data Type** list, select **String**.
   5. Select **Required**.
   6. Click **Save**.
4. At the end of the success path, add a PingOne Authentication node or an HTTP node to send a success response.

5. At the end of any failure paths, add a PingOne Authentication node or an HTTP node to send an error response.
6. Click **Save**.
7. Click **Deploy**.

Creating an Application

Create an application in DaVinci to enable your flow.

About this task

> **Note**
>
> If you want to use an existing application to launch the flow, you can start at step 5.

Steps

1. Click the **Applications** tab.
2. Click **Add Application**.

   The **Add Application** modal opens.
3. In the **Name** field, enter a name for the application.
4. Click **Create**.
5. Find the application and click **Edit**.
6. On the **General** tab, note the following parameters:
    1. Note the **Company ID**.
7. Create a flow policy:
    1. Click the **Flow Policy** tab.
    2. Click **+ Add Flow Policy**.
    3. In the **Name** field, enter a name for the flow policy.
    4. In the flow list, select your flow.
    5. In the version list, select **Latest Version**.
    6. Click **Create Flow Policy**.

       The **Edit Your Weight Distribution** modal opens.

       > **Note**
       >
       > This example only uses one flow, but if your flow policy included multiple flows or flow versions, you could use this modal to split traffic between them.
    7. Click **Save Flow Policy**.

8. Note the **Policy ID** of your flow policy.

Adding CORS Settings in PingOne

Add the domain from which the flow will be launched to the DaVinci application in PingOne to prevent CORS issues.

Steps

1. Sign on to PingOne and go to **Applications** > **Applications**.
2. Click the **PingOne DaVinci Connection** application.

   You can also add the CORS setting to any other PingOne application. For tracking and consistency, we recommend adding the CORS setting to the PingOne DaVinci application.
3. Click the **Configuration** tab.
4. Click the **Pencil** icon.
5. In the **CORS Settings** section, select **Allow specific origins**.
6. In the **Allowed Origins** field, enter the domain from which you plan to launch the flow.
7. Click **Save**.

Integration example

This example is a simple hello-world application that shows you how to specify the DaVinci API domain, your company ID, flow policy ID, and DaVinci access token, which then integrates your single-page app with the DaVinci application and associated flow policy.

> **Note**
>
> The following example won't work unless you add your region-specific information. Replace any instances of `<region>` with your regional top-level domain:
>
> - Use `.com` for North America.
> - Use `.ca` for Canada.
> - Use `.eu` for EMEA.
> - Use `.asia` for APAC.
> - Use `.com.au` for Australia.

```
<!DOCTYPE html>
<html lang="en">
  <meta charset="UTF-8" />

  <head>
    <title>Simple HTML/JS widget sample</title>
  </head>

  <body>
    <h3>Simple HTML/JS widget sample</h3>
    <br />
    <p>Widget will be displayed below</p>
    <br />
    <div id="widget" class="skWidget">Widget should appear here</div>

    <script
      type="text/javascript"
      src="https://assets.pingone.<region>/davinci/latest/davinci.js"
    ></script>
    <script>
      //*** Populate the parameters below from your DaVinci environment ***/
      const companyId = "<companyId>";
      const skApiKey = "<apiKey>";

      //*** Build the get SK Token URL. ***/
      const skGetTokenUrl =
        "https://orchestrate-api.pingone.<region>/v1/company/<companyId>/sdktoken";

      //*** Add the API Key from your DaVinci application. ***/
      var headers = new Headers();
      headers.append("X-SK-API-KEY", skApiKey);

      var requestOptions = {
        method: "GET",
        headers: headers,
        redirect: "follow",
      };

      //*** Retrieve SK Token ***/
      fetch(skGetTokenUrl, requestOptions)
        .then((response) => response.json())
        .then((responseData) => {
          var props = {
            config: {
              method: "runFlow",
              apiRoot: "https://auth.pingone.<region>/",
              accessToken: responseData.access_token,
              companyId: companyId,
              policyId: "<policyId>",
              includeHttpCredentials: true //either true or false. Set this to true to share cookies
              parameters: {
                nonce: 'string to validate'
              }
            },
            useModal: false,
            successCallback,
            errorCallback,
            onCloseModal,
          };
          /*** Invoke the Widget ****/
```

```
            console.log(props);
            davinci.skRenderScreen(
              document.getElementsByClassName("skWidget")[0],
              props
            );
          })
          .catch((error) => console.log("error", error));

        function successCallback(response) {
          console.log(response);
        }

        function errorCallback(error) {
          console.log(error);
        }

        function onCloseModal() {
          console.log("onCloseModal");
        }
      </script>
    </body>
  </html>
```

Getting the SDK token

When implementing a DaVinci application integration using the widget method, be aware that the `POST <authPath>/<companyID>/davinci/policy/<davinciFlowPolicyID>/start` request that invokes the flow takes an SDK token to authenticate. However, the call to get a DaVinci SDK token, `GET <orchestratePath>/company/<companyID>/sdktoken`, requires the application's API key to authenticate.

> **Important**
>
> The `/sdktoken` call must be executed on the server side, not in client-side code, to protect the application's API key from exposure on a public web page.

The following sample shows a server-side code snippet from a `server.js` file used to generate the DaVinci SDK token without exposing the application's API key.

> **Note**
>
> The sample won't work unless you add your region-specific information. Replace any instances of *<region>* with your regional top-level domain:
>
> - Use `.com` for North America.
> - Use `.ca` for Canada.
> - Use `.eu` for EMEA.
> - Use `.asia` for APAC.

```
/************************
* DaVinci components
************************/

// Get a Widget sdkToken
function getDVToken(cb) {
  const url = 'https://orchestrate-api.pingone.<region>/v1/company/${companyId}/sdktoken';
  fetch(url, {
    headers: {
      "X-SK-API-KEY":  <yourDavinciAppApiKey>
    },
    method: "GET"
  })
  .then(res => res.json())
  .then(data => cb(data))
  .catch(err => console.log("Error: ", err));
}
```

Using the widget with a return URL

When using the widget with social providers, PingID, or other services requiring you to supply a return URL, add coding similar to this.

Retrieve the value for a `continueToken`.

```
/**
 * Event listener for window.load()
 * Listening for a query parameter of `continueToken`
 */
window.addEventListener('load', (event) => {
  var urlParams = new URLSearchParams(window.location.search);
  if (urlParams.get('continueToken')) {
    // flush parameter from window url
    window.history.pushState({}, document.title, window.location.pathname);
    continueWidget('policyId', 'authnflow', urlParams.get('continueToken'))
  }
});
```

Invoke the widget and continue the flow using the `continueToken`.

> **Note**
>
> The following example won't work unless you add your region-specific information. Replace any instances of `<region>` with your regional top-level domain:
>
> - Use `.com` for North America.
> - Use `.ca` for Canada.
> - Use `.eu` for EMEA.
> - Use `.asia` for APAC.
> - Use `.com.au` for Australia.

```
/**
 * Recreates an instance of the Widget placed on the page,
 * and uses the provided 'continueToken' value instead of fetching a new one from DaVinci
 * @param {string} policyId - The flow policy ID for the widget to run
 * @param {string} divId - Location on the page to place the Widget
 * @param {string} continueToken - Value of the 'continueToken' query parameter
 */
function continueWidget(policyId, divId, continueToken){
  /**
   * Creates an instance of the Widget with the following:
   * @param {object} props - Properties for the Widget execution
   * @param {object} props.config - Object containing the Widget configuration
   * @param {string} props.config.method - Widget run method { "runFlow" | "continueFlow" }
   * @param {string} props.config.apiRoot - URL of the DaVinci instance for this flow
   * @param {string} props.config.accessToken - @param {string} continueToken
   * @param {string} props.config.companyId - ID of the PingOne environment that contains the flow
   * @param {string} props.config.policyId - Flow policy ID for the Widget to run
   * @param {boolean} props.useModal - Present Widget as a modal, instead of embedded
   * @param {requestCallback} props.successCallback - The callback that handles the Success response
   * @param {requestCallback} props.errorCallback - The callback that handles the Error response
   * @param {requestCallback} props.onCloseModal - The callback that handles the modal close response ('useModal' ==
true)
   */
  var props= {
    config: {
      method: 'continueFlow',
      apiRoot: 'https://auth.pingone.<region>/',
      accessToken: continueToken,
      companyId:  <companyID>,
      policyId: policyId`
    },
    useModal: false,
    successCallback, errorCallback, onCloseModal
    }
    /*** Invoke DaVinci Widget ****/
    davinci.skRenderScreen(document.getElementById(divId),props)
  }
```

Limiting the cookies passed by the widget

By default, the widget passes to DaVinci all cookies that could be presented to the origin. There can be a large number of cookies, only some of which are likely to be used for your DaVinci flows. In some cases, this can cause a `431 Error: Request Header Fields Too Large`. To avoid this possibility, you can set the `originCookies` property.

The `originCookies` property is optional and accepts a string array of cookie names. Only the cookies specified will be passed by the widget to DaVinci. Set this as you would any of the DaVinci properties.

> **Note**
>
> The following example won't work unless you add your region-specific information. Replace any instances of `<region>` with your regional top-level domain:
>
> - Use `.com` for North America.
> - Use `.ca` for Canada.
> - Use `.eu` for EMEA.
> - Use `.asia` for APAC.
> - Use `.com.au` for Australia.

```
var props= {
   config: {
     apiRoot: 'https://auth.pingone.<region>/',
     companyId:  <companyID>,
     policyId:  <policyId>,
     originCookies: ["cookie-1", "cookie-2"]
   }
}
```

To disable passing cookies from the widget, set `originCookies` to an empty array.

Launching a flow with an API call

Launch a prepared flow with an API call.

This method is effective for flows without a UI component.

Configuring the Flow

Prepare a flow to be launched with an API call.

About this task

> **Note**
>
> This procedure only covers the steps and nodes required to prepare a flow for API invocation. It assumes that you have already created a flow for the purpose you have in mind. See the Getting started with DaVinci and DaVinci Best Practices documentation for more information about building flows.

Steps

1. Click the **Flows** tab.
2. Find the flow and click **... → Edit**
3. At the end of the success path, add an HTTP node to send a JSON success response.
4. At the end of any failure paths, add an HTTP node to send a JSON error response.

5. **Optional:** If you want to pass parameters to the flow as part of the invocation, add them to the input schema.
    1. Click **Input Schema**.
    2. Click **Add** to add a new parameter.
    3. In the **Parameter Name** field, enter a name for the input parameter.
    4. **Optional:** In the Description field, enter a description for the input parameter.
    5. In the **Data Type** list, select a data type for the input parameter.
    6. Select **Required** if the parameter is required for the flow.
    7. **Optional:** Repeat steps b-f to add additional parameters.
    8. Click **Save**.
6. Click **Save**.
7. Click **Deploy**.

Creating an Application

Create an application in DaVinci to enable your flow.

About this task

> **Note**
>
> If you want to use an existing application to launch the flow, you can start at step 5.

Steps

1. Click the **Applications** tab.
2. Click **Add Application**.

   The **Add Application** modal opens.
3. In the **Name** field, enter a name for the application.
4. Click **Create**.
5. Find the application and click **Edit**.
6. On the **General** tab, note the following parameters:
    1. Note the **Company ID**.
    2. Note the **API Key**.
7. Create a flow policy:
    1. Click the **Flow Policy** tab.
    2. Click **+ Add Flow Policy**.

3. In the **Name** field, enter a name for the flow policy.
4. In the flow list, select your flow.
5. In the version list, select **Latest Version**.
6. Click **Create Flow Policy**.

> **Note**
>
> This example only uses one flow, but if your flow policy included multiple flows or flow versions, you could use this modal to split traffic between them.

7. Click **Save Flow Policy**.
8. Note the **Policy ID** of your flow policy.

Adding CORS Settings in PingOne

Add the domain from which the flow will be launched to the DaVinci application in PingOne to prevent CORS issues.

Steps

1. Sign on to PingOne and go to **Applications** > **Applications**.
2. Click the **PingOne DaVinci Connection** application.

   You can also add the CORS setting to any other PingOne application. For tracking and consistency, we recommend adding the CORS setting to the PingOne DaVinci application.
3. Click the **Configuration** tab.
4. Click the **Pencil** icon.
5. In the **CORS Settings** section, select **Allow specific origins**.
6. In the **Allowed Origins** field, enter the domain from which you plan to launch the flow.
7. Click **Save**.

Invoking the Flow

Steps

1. Using a tool such as Postman, construct the DaVinci API call using the POST method and the following URL structure:

   ```
   https://orchestrate-api.pingone.com/v1/company/<YOUR_COMPANY_ID>/policy/<YOUR_POLICY_ID>/start
   ```

   The headers should include your API key using the following format. `X-SK-API-KEY=<YOUR_API_KEY>`

   You can include one or more parameters in the call. If you do so, you must include these parameters in the input schema for the flow. The following examples include a `userId` parameter.
2. Execute the API call. The response body will be the output from the **Send Success JSON Response** action.

Examples

cURL

```
curl
--location
--request POST 'https://orchestrate-api.pingone.com/v1/company/731f6c64-619a-46e5-97a5-c7cf0be0a70e/policy/
e31c3b327523685b8e71ab9d76c83346/start' \
--header 'X-SK-API-KEY: 08cfa...45a6d'
--header 'Content-Type: application/json'
--data '{
    "userId": "b7e5ad3e-ccbd-4043-b22e-8d6d3bf8f7be"
}'
```

JavaScript and Fetch:

```
const myHeaders = new Headers();
myHeaders.append("X-SK-API-Key",
"08cfa...45a6d");
myHeaders.append("Content-Type", "application/json");

const raw = JSON.stringify({
  "userId": "b7e5ad3e-ccbd-4043-b22e-8d6d3bf8f7be"
});

var requestOptions = {
  method: 'POST',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};

fetch("https://orchestrate-api.pingone.com/v1/company/731f6c64-619a-46e5-97a5-c7cf0be0a70e/policy/
e31c3b327523685b8e71ab9d76c83346/start", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

JavaScript and NodeJS - Axios:

```
const axios = require('axios');
let data = JSON.stringify({
  "userId": "b7e5ad3e-ccbd-4043-b22e-8d6d3bf8f7be"
});

let config = {
  method: 'post',
  maxBodyLength: Infinity,
  url: 'https://orchestrate-api.pingone.com/v1/company/731f6c64-619a-46e5-97a5-c7cf0be0a70e/policy/
e31c3b327523685b8e71ab9d76c83346/start',
  headers: {
    'X-SK-API-Key': '08cfa...45a6d',
    'Content-Type': 'application/json'
  },
  data : data
};

axios.request(config)
.then((response) => {
  console.log(JSON.stringify(response.data));
})
.catch((error) => {
  console.log(error);
});
```

Launching a flow with a Ping SDK

Launch a prepared flow through an application built with one of the Ping SDKs.

The Ping SDKs provide powerful orchestration capabilities with PingOne DaVinci. They let you create flows to meet your use cases, while providing a native iOS, Android, or single-page app JavaScript experience.

This method is effective when you want complete control of your end user's experience while using a flow behind the scenes.

Configuring the Flow

Prepare a flow to be launched with the Ping SDK for Android, iOS, or JavaScript.

About this task

> **Note**
>
> This procedure only covers the steps and nodes required to prepare a flow for SDK invocation. It assumes that you've already created a flow for the purpose you have in mind. You can find more information about building flows in the Getting started with DaVinci and DaVinci Best Practices documentation.

Steps

1. On the **Flows** tab, find the flow and click **... → Edit**.
2. Verify that your user interface nodes use only compatible connectors, capabilities, and elements.

Compatible connectors and capabilities are:

- The HTTP Connector and its Custom HTML Template capability
- The PingOne Forms Connector and its Show Form capability

Compatible elements within the Custom HTML Template and PingOne Forms capabilities are:

PingOne Forms element name	Custom HTML element name	Description
Text Input	**Text field**	Lets the user enter text
Password	**Text Input** with the **Secure** setting	Lets the user enter a password that cannot be read from the screen. The PingOne Forms **Verify Password** feature is supported.
Submit Button	**Submit Button**	Lets the user submit field data and proceed
Flow Button	**Flow Button**	Lets the user trigger a new process without submitting field data
Translatable Rich Text	**Label**	Displays text for the user
Dropdown	**Dropdown**	Lets the user make a selection from a dropdown list
Radio Button List	**Radio Button List**	Lets the user make a selection from a radio button list
Checkbox List	N/A	Lets the user select any number of options from a checkbox list
Combobox	N/A	Lets the user select an existing option from a dropdown list or enter text
Social Login	**skIDP component**	Lets users sign on using a third-party identity provider (IdP)

3. Verify that your flow does not depend on any unsupported elements:
   - **SKPolling components**: SKPolling components cannot be processed by the SDK and should not be included.
   - **Images**: Images included in the flow cannot be passed to the SDK.
4. If you're using HTTP Connector nodes, ensure that any buttons that submit page data use the following parameters:
   - `type=submit`
   - The `data-skbuttonvalue` must be `Continue`, `Submit`, `Proceed`, or `Next`.

     For example:

```
<button data-id="button"
    type="submit"
    class="btn btn-primary mb-3"
    data-skcomponent="skbutton"
    data-skbuttontype="form-submit"
    data-skform="usernamePasswordForm"
    id="btnSignIn"
    data-skbuttonvalue="Submit">
Sign On
</button>
```

5. If you're using HTTP Connector nodes, ensure that any buttons that navigate to a different screen without submitting data use the following parameters:

   - The `data-skbuttonvalue` must not be `Continue`, `Submit`, `Proceed`, or `Next`.

   For example:

   ```
   <button data-id="button"
       class="btn btn-link"
       data-skcomponent="skbutton"
       data-skbuttontype="form-submit"
       data-skform="usernamePasswordForm"
       id="btnTrouble"
       data-skbuttonvalue="TROUBLE">
   Having trouble signing on?
   </button>
   ```

6. If you're using HTTP Connector nodes, add each text field, password field, and button to the node's output fields.

   These outputs are all used by the SDK, regardless of any conditional logic used to control their display in the HTML.

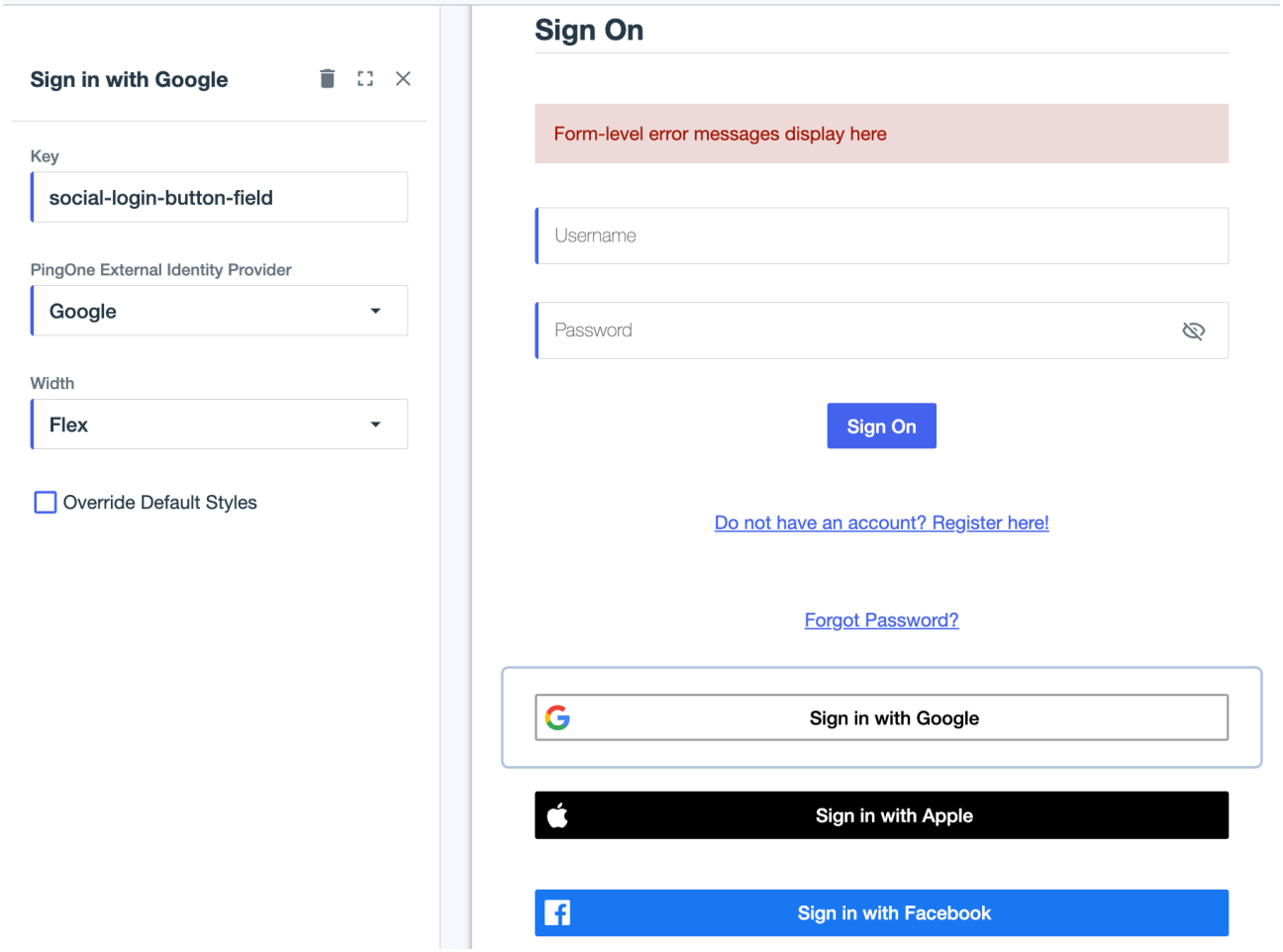
   1. Click the node.
   2. On the **General** tab, find the **Output Fields List** section, and click **Add**.
   3. In the **Property Name** field, enter a name for the property to be passed to the SDK.
   4. In the **Display Name** field, enter a name for the property to be used in the user interface.
   5. In the **Control Type** list, select **Text Field**.
   6. In the **Data Type** list, select **String**.
   7. If the value is a password, select **Secure**.
   8. In the **Value** field, select the {} option and map the field's value to the property.
   9. Repeat steps b-h for each additional output.
   10. Click **Apply**.
   11. Repeat these steps for each additional node with outputs.

7. If you're using social login, ensure that your flow uses a supported method.

Social login lets users sign on using a third-party IdP such as an Apple, Google, or Facebook. You must configure the external IdPs according to the PingOne documentation before including them in a flow.

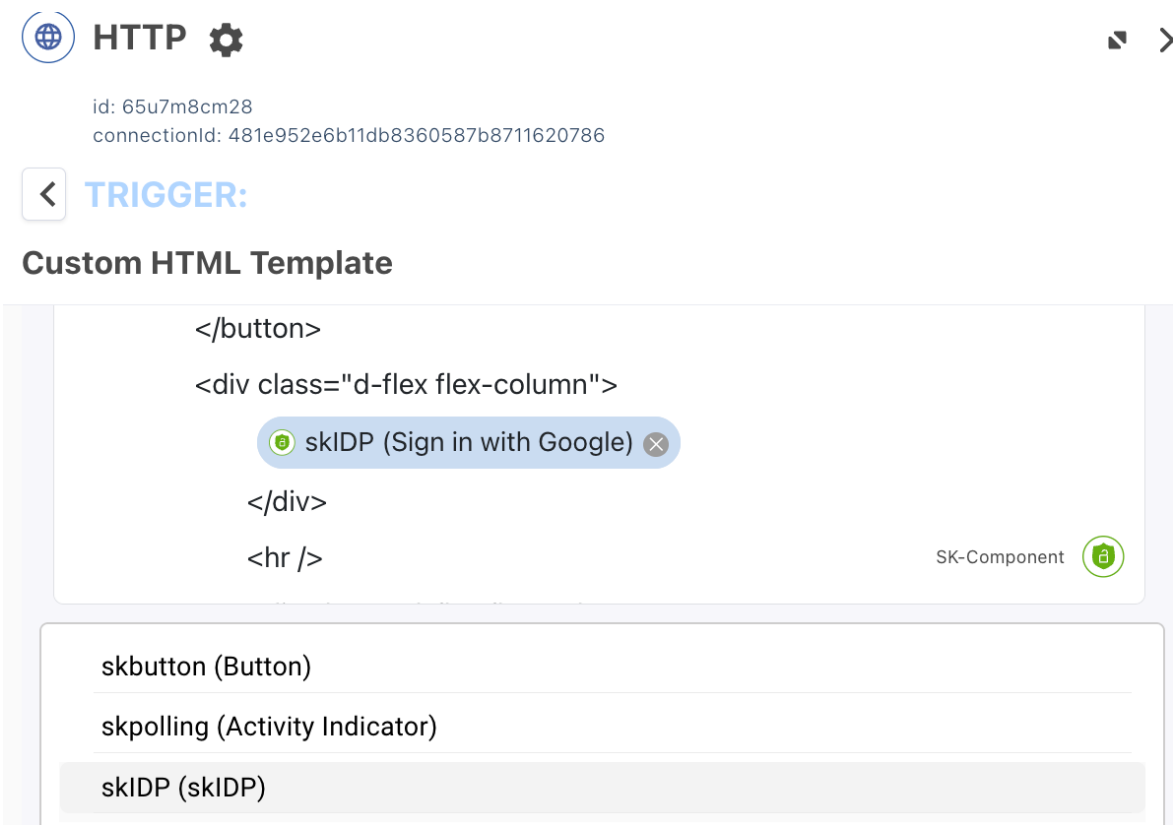
> **Note**
>
> Social Login using the **Sign On with External Identity Provider** capability of the PingOne Authentication connector is not supported in SDK flows.

- In a PingOne Form, you can use the **Social Login** tool to enable social login. Configure this button using the PingOne documentation.

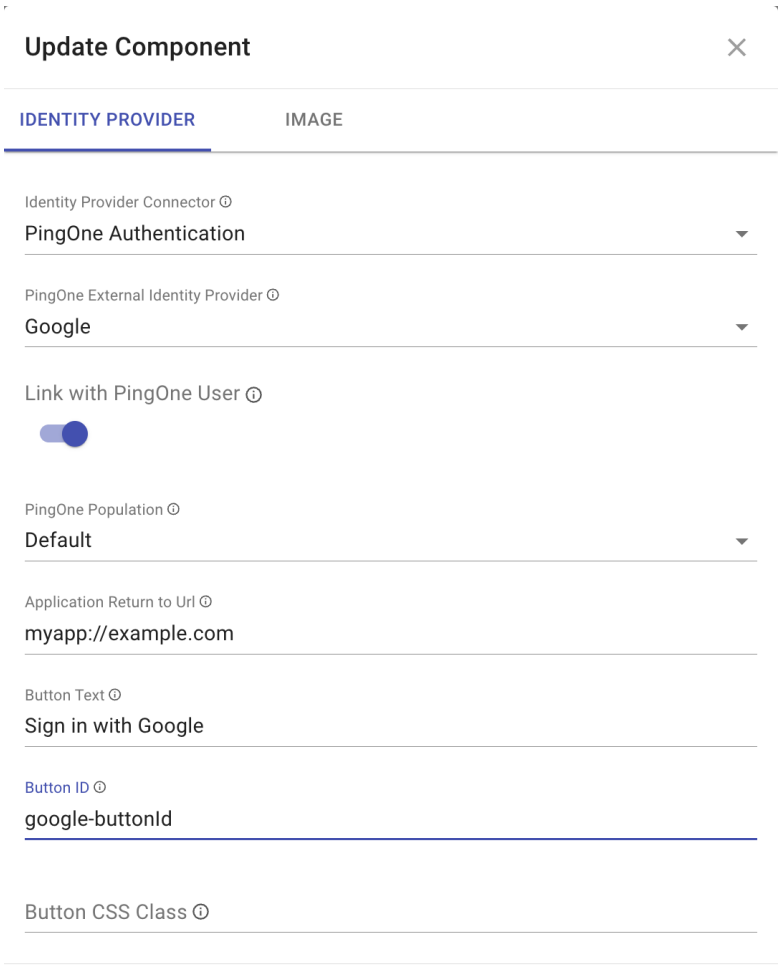


- In a Custom HTML Template, you can use the skIDP sk-component to enable social login:

  1. In the **HTML Template** section of the **General** tab, click the **{}** icon.
  2. In the **Choose Connection** list, select **SK-Component**, then select **skIDP**.
  3. Click the **skIDP** component in the **HTML Template** field.



4. In the **Identity Provider Connector** list, select **PingOne Authentication**.
5. In the **PingOne External Identity Provider** list, select a configured social login provider.
6. If you want to correlate the third-party login user with a PingOne user, select **Link with PingOne User**, then select a user population in the **PingOne Population** list.
7. In the **Button ID** field, enter an ID.



8. Click **Save**.

8. To use the user's PingOne User ID in your flow, include the `p1UserId` global parameter in your flow.

> **Note**
>
> The user's PingOne User ID is only available after the user has been identified. The user can be identified directly through PingOne or using a third-party authentication that is correlated with the user's PingOne account.

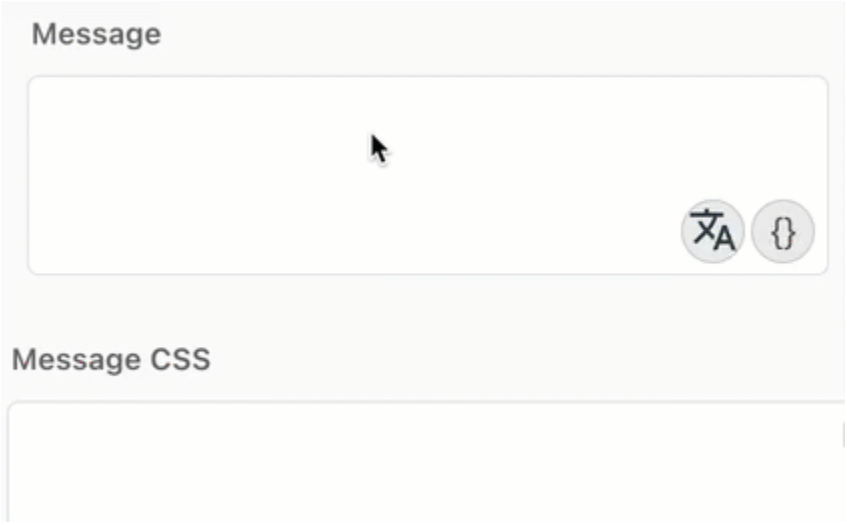
1. Open the node in which you want to include the invocation information.
2. Click the **Variables ({})** icon.
3. Click **Global**.
4. Select the **p1UserId** property.

9. To use data about the invocation method or the invocation platform type in your flow, include the `isSdk` and `platformType` global parameters in your flow.

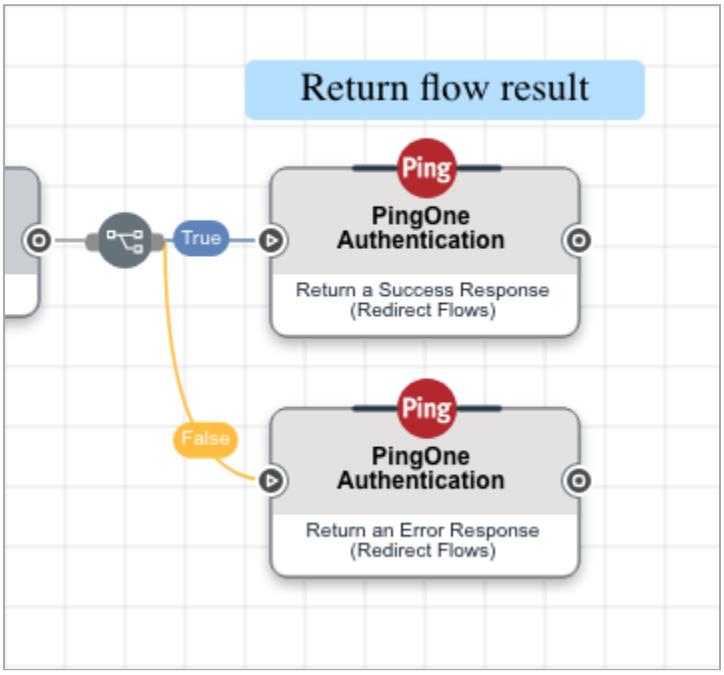
Property	Type	Description
`isSdk`	Boolean	Indicates whether the flow was launched using the SDK.
`platformType`	String	Indicates what platform was used to launch the flow if it was launched using the SDK. Valid values are `js`, `android`, and `ios`.

1. Open the node in which you want to include the invocation information.
2. Click the **Variables ({})** icon.
3. Click **Global** > **Current Request**.
4. Select the **isSdk** or **platformType** property.

   For example:



10. Click **More options (⋮) > Flow Settings** to show the flow settings.
11. Select the **PingOne Flow** option.
12. Click **Save**, then close the **Flow Settings** pane.
13. End the flow with the following two **PingOne Authentication** nodes, one for success and one for failure.



Node	Purpose
Return a Success Response (Redirect Flows)	This creates a PingOne session for the user and redirects the browser back to the source of the authentication request. This response provides the requested scopes as well as an access token or ID token.
Return an Error Response (Redirect Flows)	This redirects the browser back to the source of the authentication request. This response provides information about the error that occurred.

14. Click **Save**, then click **Deploy**.
15. If the main flow uses subflows, ensure that the subflows are not configured as PingOne flows.
    1. On the **Flows** tab, find the flow and click ... > Edit.
    2. Click **More options (⋮) > Flow Settings** to show the flow settings.
    3. Select the **PingOne Flow** option.
    4. Click **Save**, then close the **Flow Settings** pane.
    5. Repeat these steps for each additional subflow.

Creating a DaVinci application

Create an application in DaVinci to enable your flow.

Steps

1. Sign on to DaVinci.
2. On the **Applications** tab, click **Add Application**.

Result:

The **Add Application** modal opens.

3. In the **Name** field, enter a name for the application.
4. Click **Create**.
5. Find the application and click **Edit**.
6. On the **OIDC** tab, note the application parameters for the following:
    - **Company ID**
    - **Client ID**
    - **Client Secret**
    - **Issuer**
    - **Token Endpoint**
    - **JWKS Endpoint**
7. Create a flow policy:
    1. On the **Flow Policy** tab, click **+ Add Flow Policy**..
    2. In the **Name** field, enter a name for the flow policy.
    3. Select **PingOne Flow Policy**.
    4. In the flow list, select your flow.
    5. In the version list, select **Latest Version**.
    6. Click **Create Flow Policy**.

       Result:

       The **Edit Your Weight Distribution** modal opens.

       > **Note**
       >
       > This example only uses one flow, but if your flow policy included multiple flows or flow versions, you could use this modal to split traffic between them.

    7. Click **Save Flow Policy**.
    8. Note the **Policy ID** of your flow policy.

Configuring PingOne for flow invocation

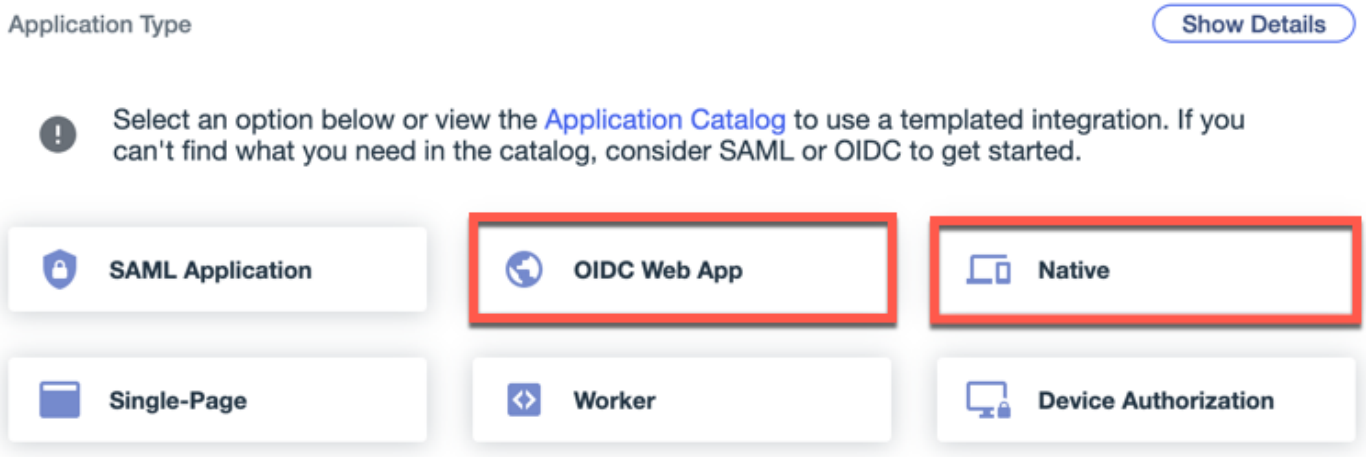
Configure an application in PingOne to launch flows.

About this task

The PingOne application is used as part of the flow invocation process.

Steps

1. Sign on to PingOne and go to **Applications** > **Applications**.
2. Click the + icon.
3. In the **Application Name** field, enter a name for the application.
4. In the **Application Type** section, select an application type corresponding to your platform:
    - If you're using the Ping SDK for JavaScript, select **OIDC Web App**.
    - If you're using the Ping SDK for iOS or Android, select **Native**.



5. Click **Save**.
6. On the **Policies** tab, click **Add Policies**.
7. On the **DaVinci Policies** tab, select one or more flow policies to add to the application.

> **Note**
>
> Only flow policies with the **PingOne Policy** option are displayed.

8. Click **Save**.
9. If you're using the Ping SDK for JavaScript, configure CORS settings for your application.
    1. On the **Configuration** tab, click the **Pencil** icon.
    2. In the **CORS Settings** section, select **Allow specific origins**.
    3. In the **Allowed Origins** field, enter the domain from which you plan to launch the flow.
    4. Click **Save**.
10. Create a revoke resource as described in the SDK documentation.
11. Register OAuth 2.0 applications as described in the SDK documentation.

Invoking the flow

To launch the flow, construct a link with the PingOne details and add it to the resource that will launch the flow. You must download the Ping SDKs and create a user experience that launches the flow.

To create the link, you need the following values:

- Client ID
- Redirect URI
- Scopes
- OIDC Discovery Endpoint

Learn more about creating the link and launching the flow in the SDK documentation.

Switching between PingOne and DaVinci widget integrations

Learn how to switch between using a flow for a PingOne redirect integration and an integration using the DaVinci widget.

> **Note**
>
> The following procedures enable you to reintegrate the same DaVinci flows for different use cases. In addition to these steps, follow the relevant procedures for your integration. For more information, see Integrating Flows into Applications.

PingOne to DaVinci widget

About this task

To change a flow integration that uses a PingOne redirect to a flow integration that uses the DaVinci widget:

Steps

1. Sign on to PingOne and go to **Applications → Applications**.
2. Remove your DaVinci flow policy from the relevant application. Click **Save**.
3. In the DaVinci console, go to **Applications** and select the application that uses the PingOne flow policy.
4. Click the **Flow Policy** tab.
5. Locate the PingOne flow policy and, in the **More options (⋮)** list, select **Delete**.
6. Go to **Flows** and select the PingOne flow.
7. Go to **More options (⋮) → Flow Settings**.
8. On the **General** tab, click the toggle to turn off **PingOne Flow**. Click **Save**.
9. Close the **Flow Settings** panel and then redeploy the flow by clicking **Deploy**.
10. Go to the **Flow Policy** tab in the application where you previously deleted the flow policy.

11. Click **Add Flow Policy** and select **All Flow Policies**.
12. Select the desired version of the previously configured flow and click **Create Flow Policy**.
13. **Optional:** Configure the **Edit Your Weight Distribution** modal.
14. Click **Save Flow Policy**.

Next steps

Integrate your flow using the DaVinci widget.

DaVinci widget to PingOne

About this task

To change a flow integration that uses the DaVinci widget to a PingOne flow integration that uses a redirect:

Steps

1. Go to **Applications** and select the application that uses the widget-based flow policy.
2. Click the **Flow Policy** tab.
3. Locate the widget-based flow policy and, in the **More options (⋮)** list, select **Delete**.
4. Go to **Flows** and select the widget-based flow.
5. Go to **More options (⋮) → Flow Settings**.
6. On the **General** tab, click the toggle to enable **PingOne Flow**. Click **Save**.
7. Close the **Flow Settings** panel and then redeploy the flow by clicking **Deploy**.
8. Go to the **Flow Policy** tab in the application where you previously deleted the flow policy.
9. Click **Add Flow Policy** and select **PingOne Flow Policy**.
10. Select the desired version of the previously configured PingOne flow and click **Create Flow Policy**.
11. **Optional:** Configure the **Edit Your Weight Distribution** modal.
12. Click **Save Flow Policy**.
13. Assign your new flow policy to the related application in PingOne.

> **Note**
>
> Learn more about assigning policies to PingOne applications in Launching a PingOne flow with a redirect.

Next steps

Integrate your flow using a PingOne redirect.

Use Cases

These use cases showcase ways in which you can use flows to solve specific business challenges.

- Creating an authentication flow
- Localizing flows with PingOne
- Adding social login with PingOne

Some solutions also combine DaVinci with PingOne or other tools to address more complex business challenges.

- PingOne for Customers Passwordless: This solution uses PingOne and a set of tailored DaVinci flows to create a registration and sign-on process that incorporates passwordless, password, and multi-factor authentication (MFA) options. Learn more in the PingOne for Customers Passwordless documentation.
- PingOne for Customers Plus: This solution uses PingOne and a set of tailored DaVinci flows to create a registration and sign-on process that incorporates a variety of MFA options. Learn more in the PingOne for Customers Plus documentation.
- Gift Card Redemption: This solution uses PingOne and a set of tailored DaVinci flows to provide a secure way for users to redeem gift cards, including step-up authentication and the ability for users to update their email address and other information. Learn more in the Gift Card Redemption documentation.
- Microsoft Entra ID: This solution lets customers use an external authentication provider for MFA through external authentication methods (EAMs). In this use case, you'll learn how to set up Entra ID, PingOne SSO, and PingID to support an EAM in Entra ID. In this scenario, Entra ID is the identity provider (IdP), and PingOne SSO and PingID are the external authentication provider. Learn more in Setting up PingOne SSO, DaVinci, and PingID as the external MFA provider for Microsoft Entra ID.

Creating an authentication flow

Create a flow to authenticate a user and optionally reset their password.

Copying the flow template

Copy the **DaVinci Flow - PingOne Sign On and Password Reset** template.

Steps

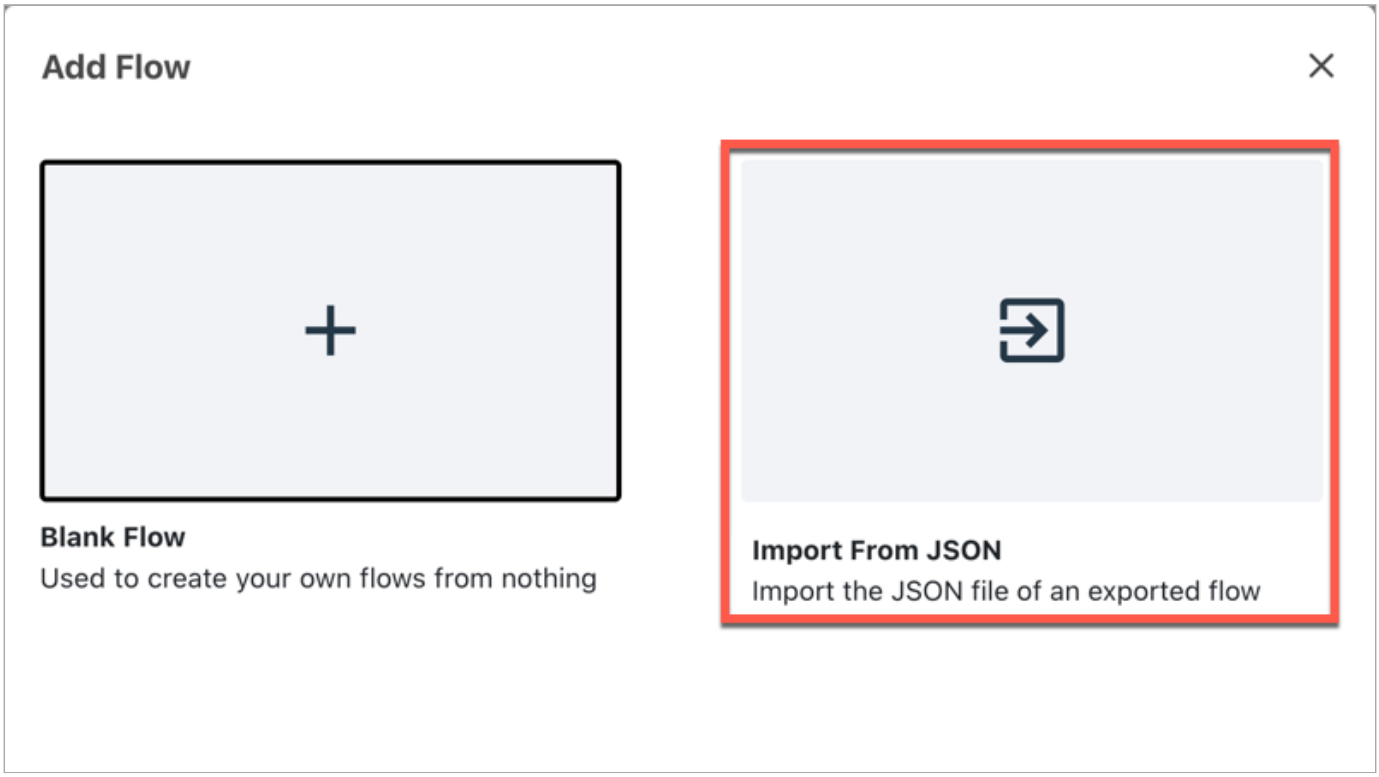
1. Go to the Integration Directory.
2. Find and open the **DaVinci Flow - PingOne Sign On and Password Reset** flow.
3. Click **Download Integration**.
4. Save the flow JSON.

Creating the flow

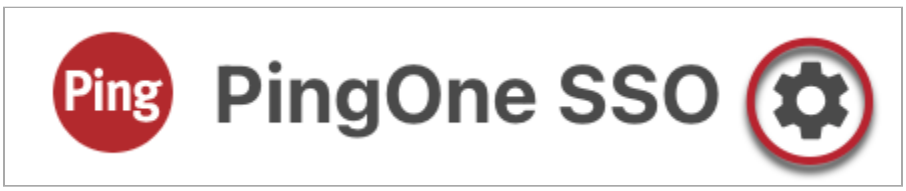
Create a new flow using the **PingOne - Sign On and Password Reset** template.

Steps

1. Sign on to your DaVinci environment.
2. Click the **Flows** tab.
3. Click **Add Flow**.
4. Click **Import From JSON** and select the saved template.



5. **Optional:** Update the name and description of the flow.
6. Click **Import**.
7. Click the **PingOne User Lookup** connector.
8. Click the **Edit Connector** icon.



9. In the **Environment ID** field, enter your PingOne environment ID.
10. In the **Client ID** field, enter your PingOne client ID.
11. In the **Client Secret** field, enter your PingOne client secret.
12. Click **Apply**.

13. Click **Save**.
14. Click **Deploy**.

Localizing flows with PingOne

Localizing a flow lets you present a single flow to customers who use multiple languages, providing a better experience for customers and reducing backend complexity.

You can use the PingOne **Languages** feature to provide translations for forms created in PingOne. In this solution, you create one or more forms in PingOne and then use the **Languages** feature to add translations for the fields, options, error messages, and other content used by those forms. You then include the forms in your DaVinci flows.

When a user reaches one of the forms, PingOne uses their browser's language setting to present them with the correct translation of the form.

Procedure

This section shows you how to create one or more forms, add translations for those forms, and add the forms to one or more flows.

> **Note**
>
> The forms, languages, and flows in your environment will vary. This procedure is as specific as possible while still applying to all environments.

Creating forms

About this task

First, create one or more forms in PingOne.

> **Note**
>
> This procedure does not cover all of the options available in the **Forms** section. You can find information about all available options in the Forms documentation.

Steps

1. Sign on to PingOne and go to **User Experience → Forms**.
2. Click the **+** icon to create a new form.
3. In the **Form Name** field, enter `User Information Example`.
4. In the **Form Usage** list, select **DaVinci Flow Forms**.
5. Click **Add Form**.
6. Select a template or a blank form as a starting point.

7. **Optional:** Add one or more custom fields and create a translation key for each one:
    1. Drag the custom field onto the form.
    2. In the left pane, click the translation key icon in the **Label** field.

    3. Select **Create New Key**.
    4. In the **Key** column, enter an internal name for the key.
    5. In the **Default Translation** column, enter the value of the key in your environment's default language.
    6. Click **Select**.
8. **Optional:** Drag one or more **PingOne Attribute Fields** onto the form.
9. Click **Save**.

Adding languages and translations

About this task

After creating your forms, enable one or more languages, then provide translations for the fields in your forms for each enabled language.

> **Note**
>
> This procedure does not discuss all options available in the **Languages** section. For more information about all available options, see the Languages documentation.

Steps

1. Sign on to PingOne and go to **User Experience → Languages**.
2. Click the **+** icon to add a language.
3. In the **Languages** list, select a language to add.
4. Click **Save Changes**.
5. Select the language that you added from the list of languages.
6. In the **Module** list, select **DaVinci Forms**.
7. Provide translations for the standard fields in your forms:
    1. In the **Page** list, select **Standard Fields**.
    2. Click the **Pencil** icon.
    3. Enter translations for the standard fields that you used in your forms.

4. Click **Save**.

8. Provide translations for the custom fields in your forms:
    1. In the **Page** list, select **Custom Messages**.
    2. Click the **Pencil** icon.
    3. Enter translations for the custom fields that you used in your forms.
    4. Click **Save**.
9. Provide translations for additional text that you used in your forms:
    1. In the **Page** list, select **Others**.
    2. Click the **Pencil** icon.
    3. Enter translations for the additional text that you used in your forms.
    4. Click **Save**.
10. **Optional:** Repeat steps 5-9 for each additional language that you have enabled.

Adding forms to flows

About this task

After adding your languages and translations for your forms, add the forms to your flow.

> **Note**
>
> This procedure does not discuss all of the options available for the PingOne Forms connector. For more information about all available options, see the Forms connector documentation.

Steps

1. Sign on to DaVinci and click the **Flows** tab.
2. Find the flow to which you want to add the forms and go to **...** → **Edit**.
3. Click the **+** icon to add a node.
4. In the **Add Connector** list, select the **PingOne Forms** connector.
5. Click and drag a line from the new node to the existing nodes.
6. Click the **PingOne Forms** node.
7. Select the **Show Form** capability.
8. In the **Form** list, select your form.
9. Click **Apply**.
10. **Optional:** Repeat steps 3-9 for each additional form that you want to add.
11. **Optional:** Repeat steps 2-10 for each additional flow.

Result

The flows now include the forms, which use the language defined in the user's browser settings. Integrate these flows into your environment using a redirect to begin using them. Learn more in Launching a PingOne flow with a redirect.

Adding social login with PingOne

Adding social login lets your users sign on using existing social accounts, simplifying their sign-on process.

You can use the PingOne **Social Login** feature to add social login buttons to forms created in PingOne. In this solution, you create one or more forms in PingOne and then add social login buttons to those forms. You then include the forms in your DaVinci flows and perform some additional configuration for user population and account matching.

When a user reaches one of the forms, they can use an existing social account to sign on.

Procedure

This section shows you how to create one or more forms, add social login buttons to those forms, and add the forms to one or more flows.

> **Note**
>
> The forms, social login options, and flows in your environment will vary. This procedure is as specific as possible while still applying to all environments.

Creating forms

Before you begin

Create an external identity provider for each social login option that you plan to use. Learn more in External IDPs.

About this task

First, create one or more forms in PingOne.

> **Note**
>
> This procedure does not cover all of the options available in the **Forms** section. Learn more about all available options in the Forms documentation.

Steps

1. Sign on to PingOne and go to **User Experience → Forms**.
2. Click the **+** icon to create a new form.
3. In the **Form Name** field, enter `User Information Example`.
4. In the **Form Usage** list, select **DaVinci Flow Forms**.
5. Click **Add Form**.
6. Select a template or a blank form as a starting point.

7. Add one or more social login options:
    1. Click the **Toolbox** tab.
    2. Drag a **Social Login** button from the left panel onto the form.
    3. Click the new **Social Login** button in the form to display its properties.
    4. In the **App Type** list, select a social login application.
    5. **Optional:** Repeat steps b-d to add additional social login options to the form.
    6. **Optional:** Update the width and style settings.
8. **Optional:** Drag one or more **PingOne Attribute Fields** onto the form.
9. Click **Save**.

Adding forms to flows

About this task

After adding social login to your forms, add the forms to your flow.

> **Note**
>
> This procedure does not discuss all of the options available for the PingOne Forms connector. Learn more about all available options in the Forms connector documentation.

Steps

1. Sign on to DaVinci and click the **Flows** tab.
2. Find the flow to which you want to add the forms and go to **... → Edit**.
3. Click the **+** icon to add a node.
4. In the **Add Connector** list, select the **PingOne Forms** connector.
5. Click and drag a line from the new node to the existing nodes.
6. Click the **PingOne Forms** node.
7. Select the **Show Form** capability.
8. In the **Form** list, select your form.
9. **Optional:** Configure the social login-specific settings:
    1. Enable **Link with PingOne User** to link the social account with the existing PingOne user.
    2. Select a **PingOne Population** for user linking, or select **Use Population ID** to specify a population ID.
    3. In the **Population ID** field, enter a population ID to use for user linking. This field is only displayed if you selected **Use Population ID** in the **PingOne Population** field.
10. Click **Apply**.

11. **Optional:** Repeat steps 3-10 for each additional form that you want to add.
12. **Optional:** Repeat steps 2-11 for each additional flow.

Result

The flows now include the forms which include social login. Integrate these flows into your environment using a redirect to begin using them. Learn more in Launching a PingOne flow with a redirect.

Connectors

Connectors form the building blocks for flows. They connect DaVinci with third parties, HTML pages, and other tools.

Each connector enables one or more capabilities that you can use as nodes in a flow. For example, a Google identity provider (IdP) connector provides the capability to sign on or get sign-on information from Google, and an HTTP connector provides the capability to present an HTML form. When you add a connector, you gain the ability to use its capabilities in your flows.

Finding connectors

Every connector requires specific information during configuration and provides a unique set of capabilities. See the Core connectors documentation and the Integration Directory for a list of connectors and their capabilities.

Core connectors

Core connectors provide the foundation for your DaVinci flows by performing basic functions.

You can also get connectors that help you integrate with other Ping Identity products and other services that your organization uses. You can search the complete list of connectors in the Integration Directory.

The following table lists and describes the core connectors and provides links to the related documentation.

Core connector	Description
Challenge	Handle asynchronous events by pausing or continuing a flow based on a transaction status.
Code Snippet	Create custom JavaScript code that you can reuse in any flow.
Cookie	Set and retrieve session cookies.
Device Policy	Check the user agent, browser information, and operating system version.
Error Message	Show customizable error messages.
FingerprintJS	Create a unique visitor ID based on browser attributes for use in fraud and analytics.
Flow Analytics	Log details about flow outcomes to be used in flow analytics.
Flow Conductor	Handle events in external systems and link to subflows.
Functions	Branch your flow using logical conditions (A > B) or based on the result of custom JavaScript code.
HTTP	Create forms and custom HTML pages or make REST API calls.
Location Policy	Check a user's IP and geographic location.
OIDC & OAuth IdP	Authenticate users with OpenID or OAuth, get user info, and create access tokens.
Screen	Display forms and customized UI to retrieve information from a user or show flow progress.

Core connector	Description
String	Create and transform string variables.
Teleport	Visually organize and subdivide a flow within the same flow canvas.
Token Management	Create and read JWT tokens and manage OIDC redirects
Variable	Store and retrieve flow and user attributes as variables.

Ping connectors

Ping connectors help you integrate your flows with other Ping Identity products and features.

You can also get connectors that help you integrate with other services that your organization uses. You can search the complete list of connectors in the Integration Directory.

The following table lists and describes the Ping connectors and provides links to the related documentation.

Ping connector	Description
PingFederate	Integrate your existing PingFederate authentication policies.
PingID	Use PingID for multi-factor authentication (MFA).
PingOne	Create and manage user accounts in PingOne, including resetting passwords and managing groups and agreements.
PingOne Authentication	Authenticate users and manage PingOne user authentication sessions. This connector is required to integrate flows into your application using the redirect and widget methods.
PingOne Authorize	Use PingOne Authorize for policy-based authorization decisions.
PingOne Credentials	Use PingOne Credentials to issue, verify, and manage digital verifiable credentials.
PingOne Forms	Show messages using your PingOne branding and themes or include forms that you create in the drag-and-drop form builder.
PingOne MFA	Use PingOne MFA for multi-factor authentication (MFA) and device enrollment.
PingOne Notifications	Send custom voice, SMS, and email notifications to cover a wide range of use cases for your customers.
PingOne Protect	Use PingOne Protect to issue MFA challenges or deny access in high-risk situations.
PingOne RADIUS Gateway	Authenticate users when accessing RADIUS clients that support the RADIUS PAP protocol.
PingOne Scope Consent	View and update user consent records.

Ping connector	Description
PingOne Verify	Use PingOne Verify to securely verify a user's identity based on a government-issued document and live face capture, also known as a selfie.

Adding a connector

Add a new connector to enable new capabilities for your flows.

Steps

1. Click the **Connectors** tab.
2. Click **Add Connector**.

   The list of available connectors is displayed. The list is divided into three sections:

   - Core Connectors (Basic flow components and standards-based integrations)
   - Ping Identity Connectors (Ping Identity products)
   - Service Connectors (Third-party platforms and services)
3. Find the connector you want to add and click the **+** icon.

   The new connector modal opens.
4. In the **Name** field, enter a name for the new connector.
5. Click **Create**.
6. Find the newly-created connector in the list of your connectors and click **... → Edit**.
7. Update the connector's properties.

   Each connector has different properties. See the connector's reference documentation for more information about required and optional properties.
8. Click **Apply**.

Editing a connector

Edit an existing connector to update its properties.

Steps

1. Click the **Connectors** tab.
2. Find the connector in the list of your connectors and click **... → Edit**.
3. Update the connector's properties.

Each connector has different properties. See the connector's reference documentation for more information about required and optional properties.

4. Click **Apply**.

Renaming a connector

Rename a connector to display a new name for the connector in the user interface.

Steps

1. Click the **Connectors** tab.
2. Find the connector and click **... → Rename**.

   The **Rename Connector** modal opens.
3. In the **Name** field, enter a new name for the connector.
4. Click **Rename**.

Cloning a connector

Clone an existing connector to create a new connector with the same settings.

Steps

1. Click the **Connectors** tab.
2. Find the connector and click **... → Clone**.

   A new connector is created with the same type and settings as the original, with a suffix indicating that it was cloned.
3. **Optional:** Edit the new connector as described in Editing a connector.

Deleting a connector

Delete a connector to permanently remove it.

About this task

> **Note**
>
> Verify that the connector is not being used in any active flows before deleting it.

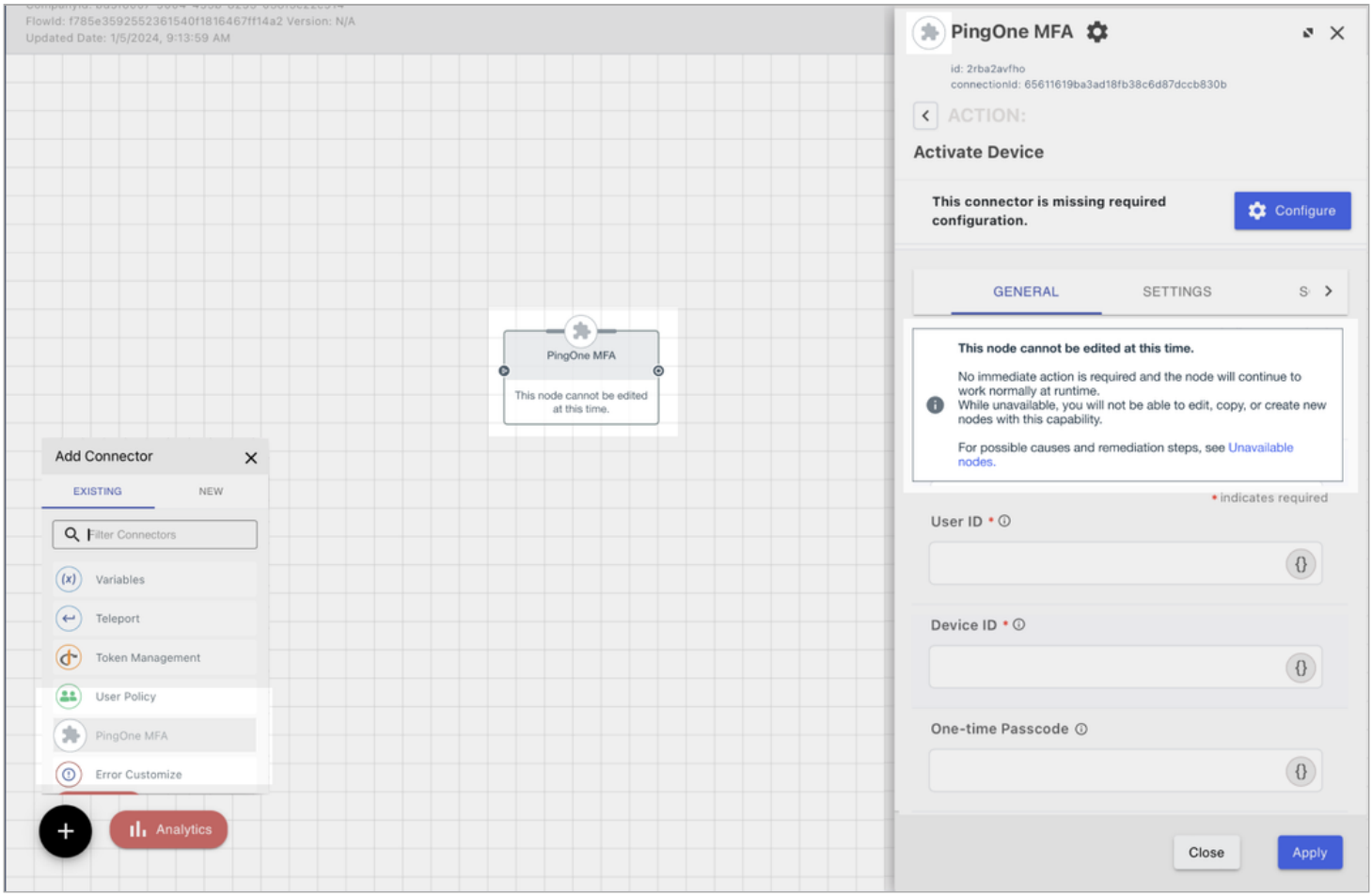
Steps

1. Click the **Connectors** tab.
2. Find the connector and click **... → Delete**. On the confirmation modal, click **Delete**.

Understanding and troubleshooting unavailable capabilities

When a capability is unavailable, nodes that use that capability still function during runtime but cannot be edited or added.

If a capability is unavailable in your environment, nodes that use that capability are indicated with warning text in the flow canvas. These nodes function normally when the flow is run so that the runtime behavior for your users isn't degraded, but they cannot be edited, updated, or added while the capability is unavailable.



Causes

Multiple conditions can cause a capability to be unavailable:

- The flow was exported from an environment with access to a connector or capability and imported into an environment without the same access. This can occur if the connector or capability is in early access or requires a specific license or service.
- The connector or capability is temporarily unavailable because of administrative portal issues.
- The connector or capability is being deprecated.

Troubleshooting unavailable capabilities

If you have one or more nodes with unavailable capabilities in your environment, there are multiple troubleshooting steps you can take:

1. If you imported the flow from another environment, make sure that the current environment has the same licensing and services as the original environment.
2. Review the connector's release notes or integration directory entry to see if it is being deprecated.
3. Wait for 2 to 5 days to see if the issue resolves itself.
4. Contact Ping support.

Available actions

When a capability is unavailable, you can take these actions:

- Manipulate the node on the canvas:
  - Move the node
  - Change the lines leading into or out of the node
- Use right-click options on the node:
  - **Clone** (the cloned node has the same limitation)
  - **Copy** (the pasted node has the same limitation)
  - **Replace**
  - **Change Linked Connector**
  - **Disable**
  - **Delete**
- Open the node properties to:
  - See the node configuration
  - See the deprecation message if one is present
  - Open the capabilities list to select another capability from the same connector

Disabled actions

When a capability is unavailable, you cannot:

- Edit the capability configuration
- Add a new node that uses the capability

Using connectors securely

While Ping Identity provides many proprietary integrations for PingOne DaVinci, some connectors work with third-party services. You should review the security best practices documentation for those services.

Some general security best practices to consider when using third-party connectors in your DaVinci flow are:

- When passing any secrets, keys, or passwords as output variables through the HTTP connector, mark them as `Secure` in the connector configuration.
- The account with the third-party service or on-premise resource should follow the principle of least privilege and only be granted the permissions necessary to perform the actions required by the connector.
- Whenever using custom JavaScript, HTML, or CSS in a DaVinci connector, you should follow general secure coding guidelines to avoid the introduction of any security vulnerabilities, privacy violations, or other unintended behavior.

Changing the logging level

Some connectors can process a user's personally identifiable information (PII) such as name, address, email, and birthdate. To prevent inadvertent logging of any sensitive user data, you should not enable debug logging in any production-level flows that use connectors that can process PII.

About this task

To view and change the logging level for your DaVinci flow:

Steps

1. Click the **More Options (⋮)** icon and select **Flow Settings**.
2. On the **Logging** tab, view the **Log Level** list.
3. If the current selection is **Debug**, select **Info**.
4. Click **Save**.

Result

You can now see if your flow is in Debug mode and disable debug logging. For more information, see Debugging and analytics.

Marking output fields as secure

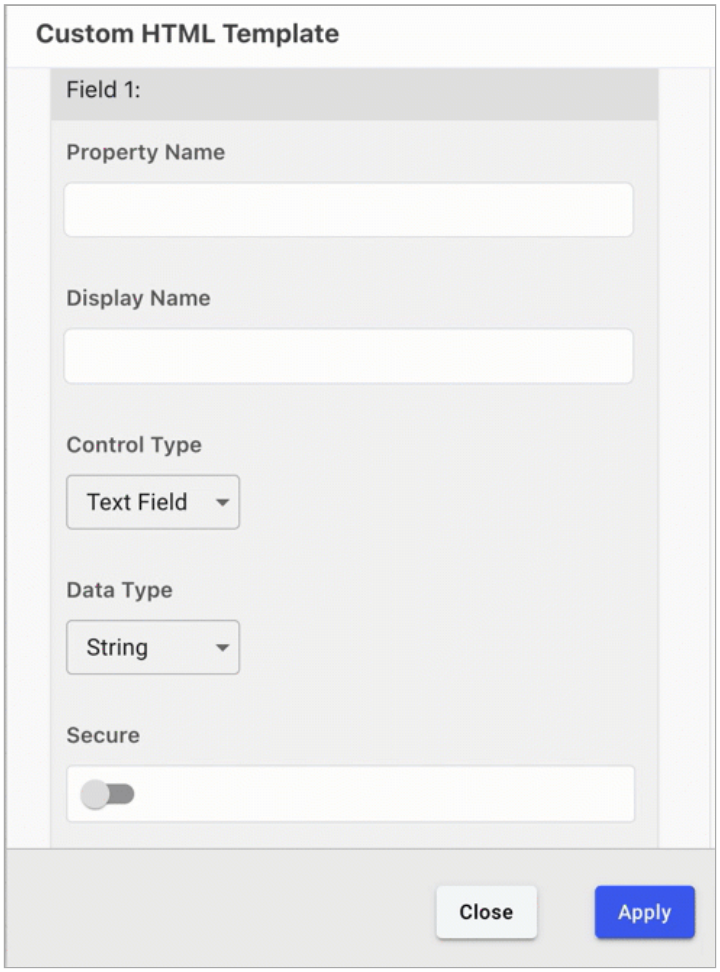
You should mark output fields secure when adding custom output fields for a connector such as the HTTP connector.

About this task

To mark output fields as secure in the **Custom HTML Template** of the HTTP connector:

Steps

1. Add an HTTP connector in DaVinci.
2. Complete the **Property Name** and **Display Name** fields.



3. Click the **Secure** toggle and click **Apply**.

Result

The output field is now marked as secure, which acts as an additional safeguard against the logging of any sensitive PII.

Forms

Forms is a PingOne feature that lets you build custom forms for use in DaVinci flows.

You can click the **Forms** tab to launch the feature in PingOne. From there, you can build custom forms that will then be available for use in DaVinci. Learn more about building forms in PingOne in the Forms documentation.

You can use the **Languages** tab to create text for your forms in multiple languages. Learn more about language features in the Languages documentation.

You can use the **Social Login** tool to enable social login for your users. Learn more about using this feature in Adding social login with PingOne.

The branding and themes for the form can also be customized through PingOne. Learn more about branding and themes in the Branding and Themes documentation.

After you create a form in PingOne, you can include it in a flow using the PingOne Forms connector. Learn more about using this connector in the DaVinci Forms Connector documentation.

You can also create and manage forms through an API. Learn more about using the API in the Forms API documentation.

UI Studio

The UI Studio lets you create user interface templates that match your company style and branding, which you can include in flows using an HTTP connector.

Creating a UI template

Create a UI template to control the UI design inside your flows.

Steps

1. Click the **UI Studio** tab.
2. Click **Add UI Template**.
3. Enter a **Name** for the UI template.
4. **Optional:** In the **Description** field, enter a description for the UI template.
5. Find the newly created UI template and click **Edit**.
6. **Optional:** Enter HTML for your UI template:
    1. Click the **HTML** tab.
    2. In the **Template (HTML)** field, enter the HTML for your UI.
    3. In the **Form Validation Rules** section, click **Add** to add one or more properties to evaluate.
    4. In the **Property Name** field, enter a property name.
    5. Click **Add** to add one or more rules for the property.
    6. For each rule, select a rule in the **Rule Name** list.
    7. For each **Required** rule, enter a **Validation Message** that displays if the required property is not present.
    8. For each **Email** rule, enter a **Validation Message** that displays if the property is not a valid email.
    9. For each **Length** rule, enter a **Minimum**, **Maximum**, or **Exact** value for the property and a **Validation Message** that is displayed for each length restriction that isn't met.
    10. For each **Format** rule, enter the **Regex** that defines the required format and the **Validation Message** that displays if the property doesn't match the format.
    11. For each **Equality** rule, enter the **Other Property** that this property must match, and the **Validation Message** that displays if the properties don't match.
7. **Optional:** Enter CSS for your UI template:
    1. Click the **CSS** tab.
    2. In the **Style (CSS)** field, enter the CSS for your UI.
8. **Optional:** Enter a script for your UI template:
    1. Click the **Script** tab.

2. In the **Script (JavaScript)** field, enter the script for your UI.

9. **Optional:** Enter a schema for your UI template:
    1. Click the **Schema** tab.
    2. In the **Input Schema** field, enter the input schema for your UI.
    3. In the **Output Schema** field, enter the output schema for your UI.
10. Click **Apply**.

Editing a UI template

Edit an existing UI template.

Steps

1. Click the **UI Studio** tab.
2. Find the UI template and click **Edit**.
3. Update the desired fields in the UI template.
4. Click **Apply**.

Deleting a UI template

Delete an existing UI template.

Steps

1. Click the **UI Studio** tab.
2. Find the UI template and click **Delete**.
3. In the confirmation window, click **Delete**.

Applications

An application in DaVinci represents one of your applications and gives you fine-grained control over which flows can be run through that application and by what methods.

An application acts as a gateway between your site and the flows you have created in DaVinci. The application contains settings to determine how external sites can send requests for flows, what flows can be requested, and how users and resources from other sites are managed. External sites can only run flows that are made available through an application.

The **General** and **OIDC** tabs let you configure receivers for incoming requests. Other tools can send requests to launch flows using API endpoints described in the **General** tab or the OpenID Connect (OIDC) parameters in the **OIDC** tab.

The **Flow Policy** tab lets you control which flows are run through the application. A flow policy is an entity that points to one or more flows or versions of flows. You can use a flow policy to make sure that a specific version of a flow, such as the latest version, is always used. You can also split traffic between different flows or flow versions for A/B testing or other purposes.

The **Connections** tab lets you direct requests to specific connectors.

Creating an application

Create a new application to enable DaVinci to interact with that application.

Steps

1. Click the **Applications** tab.
2. Click **Add Application**.

   The **Add Application** modal displays.
3. In the **Name** field, enter a name for the application.
4. Click **Create**.
5. Find the application and click **Edit**.
6. Update the application's properties.

   > **Note**
   >
   > Although an application doesn't require most properties, you must configure a flow policy in the **Flow Policies** tab before you can run flows through the application. See Configuring a flow policy for more information.
7. Click **Apply**.

Configuring general application settings

Configure an application's general properties and API settings.

Steps

1. Click the **Applications** tab.
2. Find the application and click **Edit**.

3. Click the **General** tab.
4. **Optional:** In the **Name** field, update the name of the flow.
5. **Optional:** Note the **Company ID** and **Client ID**.

> **Note**
>
> These fields are read-only. You can copy the values if necessary.

6. **Optional:** Click the **Enable API Key** toggle to enable the use of an API key for API calls.
7. **Optional:** Click **Regenerate API Key** to generate a new API key.

   Generating a new API key invalidates the previous key.
8. **Optional:** Reveal and copy the **API Key** for use in API calls.
9. **Optional:** In the **FIDO2 Connection** list, select a FIDO2 authentication method to be used for `/credentials/fido2` API paths.

> **Note**
>
> This feature is currently in limited release. To request access to this feature, open a support case.

10. Click **Apply**.

Configuring OIDC settings

Configure OpenID Connect (OIDC) for your application to enable it as a method for invoking flows.

Steps

1. Click the **Applications** tab.
2. Find the application and click **Edit**.
3. Click the **OIDC** tab.
4. **Optional:** Click **Regenerate Client Secret** to create a new client secret.

   Generating a new client secret invalidates the previous secret.
5. Reveal and copy the **Client Secret** for use in OIDC connections.
6. In the **Redirect URLs** field, enter one or more redirect URLs for the application.
7. In the **Logout URLs** field, enter one or more logout URLs for the application.
8. In the **Scopes** section, select one or more scopes based on your application's needs.

Option	Description
OIDC	Select the **OIDC** check box.

Option	Description
Profile	Select the **Profile** check box.
Flow Analytics	Select the **Flow Analytics** check box.

9. In the **Grants** section, select one or more grants based on your application's needs.

Option	Description
Authorization Code	Select the **Authorization Code** check box.
Implicit	Select the **Implicit** check box.
Client Credentials	Select the **Client Credentials** check box.

10. Copy the **Issuer**.
11. Copy the **Token Endpoint**.
12. Copy the **UserInfo Endpoint**.
13. Copy the **JWKS Endpoint**.
14. Copy the **JWKS**.
15. Click the **Enforce receiving signed requests?** toggle to require that incoming requests be signed.
16. Provide a method for verifying service provider JSON web key sets (JWKS):

    Choose from:

    - In the **Service Provider (SP) JWKS URL** field, enter a URL.
    - In the **Service Provider (SP) JWKS Keys to Verify Authorization Request Signature** field, enter one or more keys.

17. Click **Apply**.

Flow Policies

Flow policies let you control which flows are launched through the application.

Choose from one of the following topics:

- Configuring a flow policy
- Editing a flow policy
- Deleting a flow policy
- Flow policy metrics

Configuring a flow policy

Configure flow policies to control which flows and flow versions are displayed for users.

Steps

1. On the **Applications** tab, browse or search for the application and click **Edit**.
2. On the **Flow Policy** tab, click **+ Add Flow Policy**.
3. In the **Policy Name** field, enter a name for the flow policy.
4. Select **PingOne Flow Policy** to enable flows in the policy to be launched directly through PingOne.

   This option cannot be changed after the flow policy is created. PingOne flow policies can only include flows and flow versions that have the **PingOne Flow** setting enabled.
5. (Optional) If the flow policy is a PingOne flow policy, select one or more options for skipping the flow if an existing session is found.

   This option is useful if you want to avoid unnecessary flow executions for users who already have a session.

   1. Select **Password Based Authentication** to skip the flow if a password-based session exists, and select **MFA Based Authentication** to skip the flow if an MFA-based session exists.
   2. For each selected authentication method, provide a time range by entering a number and selecting minutes, hours, or days. The flow is only skipped if the existing session was created within the time range you select.
6. Click **Next**.
7. Add one or more flows to the policy:

   1. In the **Flows** section, select a flow.
   2. In the **Version** section, select one or more versions of the flow to use.

      The **Latest Version** option always uses the latest version.
   3. (Optional) Repeat the previous steps to add additional flows.
8. Click **Next**.

   Result:

   The **Edit Your Weight Distribution** modal opens.
9. Add weight distribution and analytics information for each flow and flow version:

   1. In the **Distribution** field for each flow version, enter or select a distribution weight from 1 - 100.

      When a flow policy with more than one flow is invoked, the flow policy selects a flow to run, using the distribution weight for each flow as the percent chance of its selection. You can use this feature to A/B test flows or flow versions.
   2. (Optional) Click **Add IP Whitelist**, then enter one or more IP addresses in the **Whitelist IP** field.

> **Note**
>
> If a request comes from an IP address on the allow list, the weight is ignored, and the specified flow is triggered.

3. (Optional) In the **Analytics - Select Success Nodes** list, select one or more nodes that, when run, indicate that the flow run was successful.

This information is used to calculate the flow policy's success rate.

10. Click **Create Flow Policy**.
11. Click the **General** tab, then click **Apply**.

Editing a flow policy

Edit existing flow policies to update which flows and flow versions are displayed for users.

Steps

1. On the **Applications** tab, browse or search for the application and click **Edit**.
2. On the **Flow Policy** tab, locate the flow policy and, in the **More options (⋮)** list, select **Edit**.
3. (Optional) In the **Name** field, update the name of the flow policy.
4. (Optional) If the flow policy is a PingOne flow policy, select one or more options for skipping the flow if an existing session is found.

   This option is useful if you want to avoid unnecessary flow executions for users who already have a session.

   1. Select **Password Based Authentication** to skip the flow if a password-based session exists, and select **MFA Based Authentication** to skip the flow if an MFA-based session exists.
   2. For each selected authentication method, provide a time range by entering a number and selecting minutes, hours, or days. The flow is only skipped if the existing session was created within the time range you select.
5. Click **Next**.
6. (Optional) Update the flows and flow versions included in the flow policy:
   1. Add a flow by selecting it in the **Flows** section, then selecting one or more versions in the **Version** section.
   2. Update the included versions for a flow by selecting it in the **Flows** section, then adding or removing versions in the **Version** section.
   3. Remove a flow by clicking the **X** icon in the **Flows Added** section.
7. Click **Next**.
8. (Optional) Update the weight distribution and analytics information for each flow and flow version:
   1. In the **Distribution** field for each flow version, enter or select a distribution weight from 1 - 100.
   2. (Optional) Click **Add IP Whitelist**.

3. (Optional) In the **Whitelist IP** field, enter one or more IP addresses.

> **Note**
>
> If a request comes from an IP address on the allow list, the weight is ignored, and the specified flow is triggered.

4. (Optional) In the **Analytics - Select Success Nodes** list, select one or more nodes that, when run, indicate that the flow run was successful.

This information is used to calculate the flow policy's success rate.

9. Click **Update Flow Policy**.

Deleting a flow policy

Delete an existing flow policy to permanently remove it from use.

Before you begin

Verify that the flow policy is not in use.

Steps

1. Click the **Applications** tab.
2. Find the application and click **Edit**.
3. Click the **Flow Policy** tab.
4. Locate the flow policy and in the **More options (⋮)** list, select **Delete**.

   A confirmation modal opens.
5. Click **Delete**.

Flow policy metrics

The **Flow Comparison Metrics** page lets you view data about the flows included in a flow policy. You can view it by locating the flow policy and, in the **More options (⋮)** list, selecting **Comparison Metrics**.

Data	Description
Flow Variation Names	The name and version of the flow.
Total Visitors	The number of runs of the specified flow and version.
Success Rate	The percentage of flow runs that included one or more of the success nodes as defined in the flow policy.

Configuring connection settings

Add connections to an application to enable direct interactions with that connector through REST API calls.

About this task

> **Note**
>
> This feature is currently in limited release. To request access to this feature, open a support case.

Steps

1. Click the **Applications** tab.
2. Find the application and click **Edit**.
3. Click the **Connections** tab.
4. In the **Add Connection** list, select one or more connections.
5. Click **Apply**.

Deleting an application

Delete an application to remove it from the UI and stop all flow use for it.

About this task

> **Warning**
>
> Verify that the application is not being used in an active production environment.

Steps

1. Click the **Applications** tab.
2. Find the application and click **Delete**.

   A confirmation message displays.
3. Click **Delete**.

Variables

Variables let you track and adjust values from within your flows.

Variables are values that can be read and modified during a flow. They can be strings, Boolean values, numbers, or objects.

You can use variables in a flow to track information, and you can use a variable's value to determine which nodes are run within the flow.

Every variable has a context, which determines how widely its value is shared:

Company Context

The variable has a single value for the company. This value is used in all flows and for all users.

Company context variables can be created as **Secret** variables, which are designed to store sensitive values such as API keys, client secrets, and access tokens. The current value for secret variables cannot be seen in the UI, logs, analytics, exported flows, or API queries.

> **Note**
>
> Secret variables can only be used in the HTTP connector's Make REST API Call capability in the **Headers** and **Body** parameters.

Flow Context

The variable is tied to a specific flow and has a single, persistent value until that value is changed. The current value can be viewed in the **Variables** tab.

To change variable values, you can run the containing flow or update the variable on the **Variables** tab. The variable's new value is set to the value of the latest update using either method.

Flow Instance Context

The variable can be used in multiple flows.

- If the variable's value is set within a flow, the variable instance in that flow gets the value set by the flow's execution.
- If the variable's value is not set within a flow, the variable instance in that flow inherits the value set on the **Variables** tab.

User Context

The variable has a separate value for each user. If you use a variable with this context in a flow, the user must be identified.

Adding a variable

Add a new variable that can be used in flows.

Steps

1. Click the **Variables** tab.

2. Click **Add New Variable**.
3. In the **Name** field, enter a name for the variable.

> **Note**
>
> The name `challenge` is reserved and cannot be used.

4. In the **Variable Context** list, select a context in which the variable is shared:

   Choose from:

   - **Flow Instance Context**: The variable has one value for each instance of each flow.
   - **User Context**: The variable has one value for each user. The user must be identified.
   - **Company Context**: The variable has one value for the company, across all flows and users. All secret variables use this context.

     > **Note**
     >
     > A variable with a flow context can only be created within a flow by using the Variables connector. After you create a flow context variable, you can edit its value in the **Variables** tab.

5. **Optional:** In the **Description** field, enter a description for the variable.
6. In the **Data Type** list, select a data type:

   Choose from:

   - **String**
   - **Boolean**
   - **Number**
   - **Object**
   - **Secret**: This data type indicates that the variable value is sensitive and should not be visible in the UI, analytics, logs, or API queries. It is only available for company context variables. Secret variables are for custom API calls, so they can only be used in the **Headers** and **Body** parameters of the Make REST API Call capability (HTTP connector). They are not available anywhere else in DaVinci.

7. In the **Value** field, enter an initial value for the variable.
8. **Optional:** Click the **Mutable** toggle to allow nodes within a flow to change the value of the variable.

   You can trace the expected value of a variable through the nodes of your flow to help you decide whether the variable should be mutable or not.

9. **Optional:** In the **Min** field, enter a minimum value for the variable.
10. **Optional:** In the **Max** field, enter a maximum value for the variable.
11. Click **Create**.

Editing a variable

Edit an existing variable to change its properties.

About this task

> **Note**
>
> You cannot change the name and context for a variable.

Steps

1. Click the **Variables** tab.
2. Find the variable you want to edit and click the **Pencil** icon.
3. **Optional:** In the **Description** field, update the description for the variable.
4. **Optional:** In the **Data Type** list, select a data type:

   Choose from:

   - **String**
   - **Boolean**
   - **Number**
   - **Object**
   - **Secret**

     > **Note**
     >
     > If you change a variable's data type from **Secret** to another type, the existing value is lost.

5. **Optional:** In the **Value** field, enter a value for the variable.

   For flows with the **Company Context**, this field updates the current value. For all other contexts, this field sets the initial value.

   If the variable's data type is **Secret**, the existing value can only be replaced. It cannot be edited or viewed.

6. **Optional:** Click the **Mutable** toggle to allow nodes within a flow to change the value of the variable.
7. **Optional:** In the **Min** field, enter a minimum value for the variable.
8. **Optional:** In the **Max** field, enter a maximum value for the variable.
9. Click **Update**.

Deleting a variable

Delete an existing variable to remove it from the UI and from all flows.

About this task

> **Note**
>
> If you plan to delete a variable used in one or more flows, you should update the flows to remove the variable before deleting it.

Steps

1. Click the **Variables** tab.
2. Find the variable you want to delete and click the **Delete** icon.

   A confirmation message displays.
3. Click **Delete**.

Using variables in flows

You can use variables in multiple ways within a flow.

Variables connector

You can add the Variables connector to your flows. This connector has multiple capabilities that let you add variables to a flow and increment or otherwise change the variable's value.

The variables connector can be used to update User variables and Flow Instance variables. It can also be used to create flow instance variables, but user variables must be created in the variables tab.

Including variables in code

You can include variable values in the code fields of nodes such as HTTP nodes, or as inputs in the fields of other nodes. Within the code field, click **{}**, then select a variable type.

You can also directly include a variable using the following syntax:

Company Variable

```
{{global.company.variables.variableName}}
```

User Variable

```
{{global.userInfo.variables.variableName}}
```

Flow Variable

```
{{global.flow.variables.variableName}}
```

Flow Instance Variable

```
{{global.variables.variableName}}
```

When you use a variable in this way, the variable's exact value is included in the code, and you should take the variable type into account when writing the code. For example, a string variable should be surrounded by quotation marks, while a boolean variable or number variable should not.

Sharing variable values between flows

You can pass variable data between flows and subflows using input and output schemas, which pass information into and out of the subflow.

To use an input schema in a subflow, open the flow and click **Input Schema**. Use the variable name the **Parameter Name**, select the **Data Type** that corresponds to the variable data type, and enable **Required** if the variable is necessary for the flow to function. After you add a parameter to a subflow's input schema, the Invoke Subflow or Invoke UI Subflow node that launches that subflow will include input fields that correspond to the input schema parameters. Add the variable values to these fields using the syntax described above.

To use an output schema in a subflow, click ⋮ → **Output Schema**, then declare the variable's name and type as part of the output schema JSON.

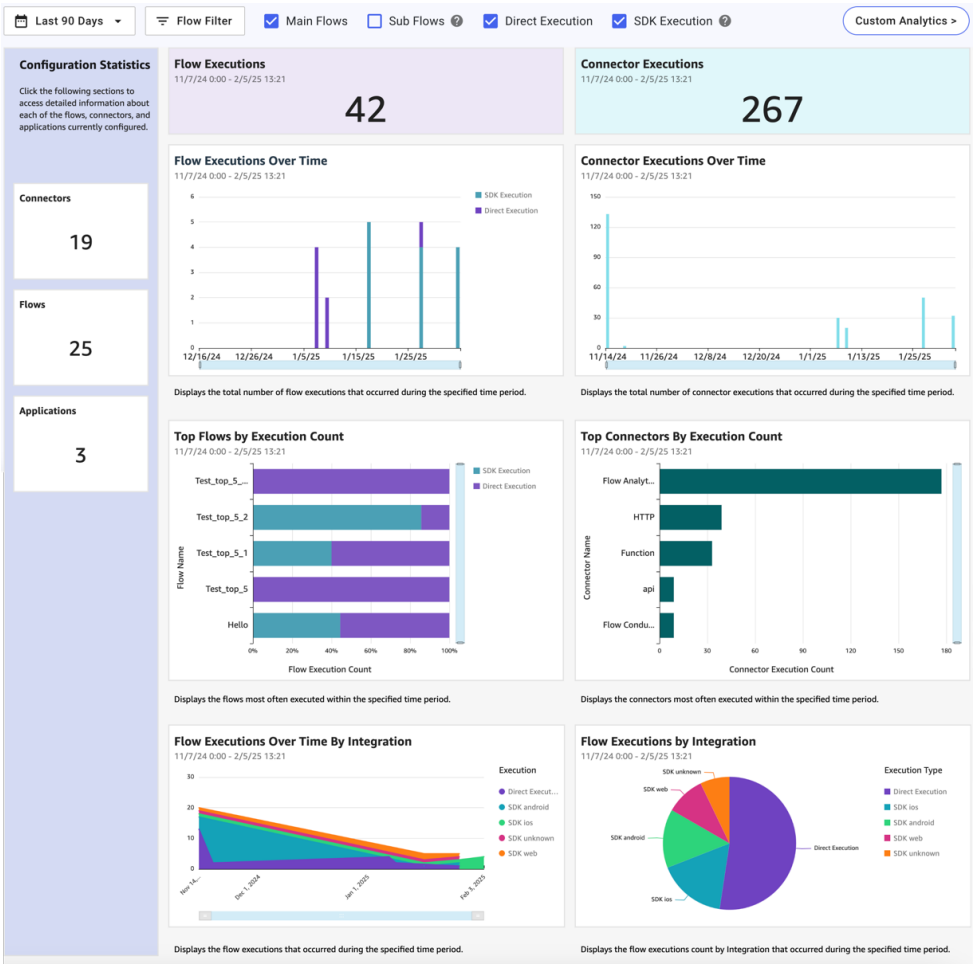
Next, open each of the nodes that sends a JSON success, error, or custom response at the end of the subflow. In the **Additional Fields in the Response** section, click + **Field**. In the **Value** field, click **{}** and select Flow Instance Variables, then click the variable you want to share. You can also use the syntax described above to include variable values. Apply your changes to the node.

Dashboard

On the PingOne DaVinci dashboard, use the graphs to view your current end-user activity levels.

Overview

Hover over data points to see specific information or click the **Maximize** ( ) icon for an individual graph to expand the appearance.



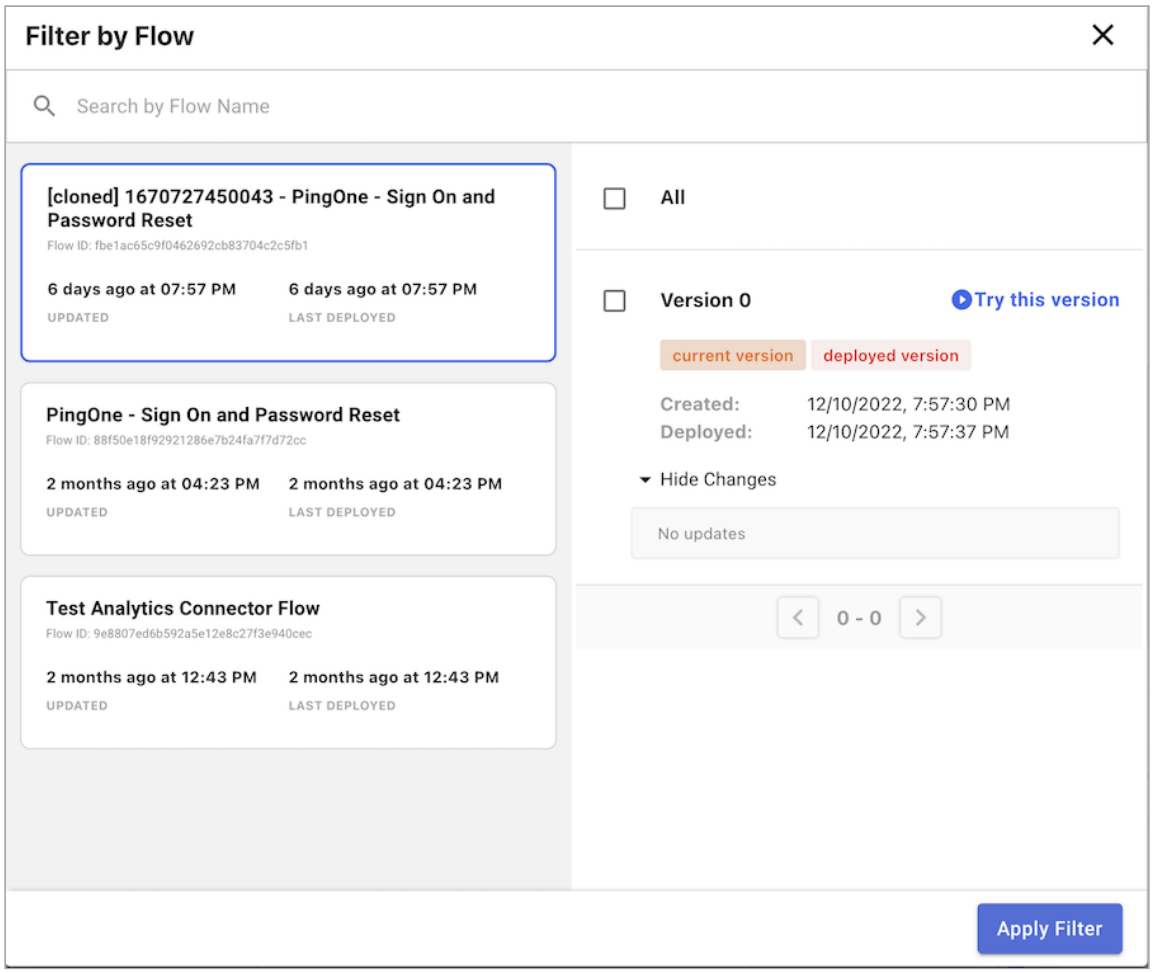
Use the checkbox options to filter the flow executions that are included in the flow statistics:

- **Main Flows** includes flows that are not invoked by another flow.
- **Sub Flows** includes flows that are invoked by another flow.
- **Direct Flows** includes flows that are invoked using a redirect, the widget, or an API call.
- **SDK Flows** includes flows that are invoked using the SDK.

The connector statistics ignore these selections and count all connector executions across all flows.

Click the date picker to select date ranges or create custom ones. Click and drag the horizontal and vertical sliders to adjust the ranges. The currently selected data range is displayed at the top of each section.

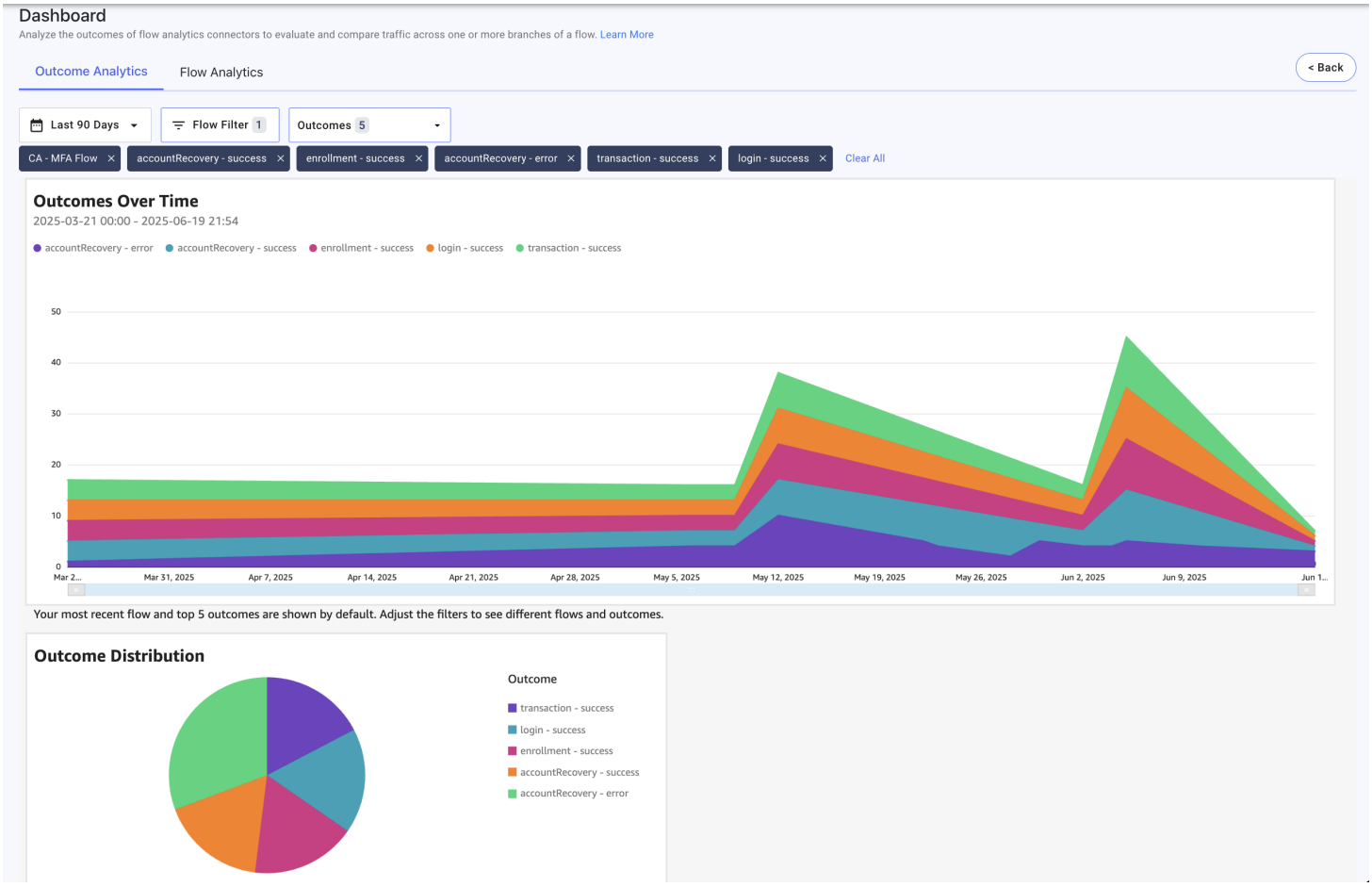
Click **Flow Filter** to open the **Filter by Flow** window. Here you can select specific flows and versions of flows to filter the data in the graphs by.



You can also use the following icons to adjust any changes to the dashboard.

Icon	Description
	Reset to original dashboard.
	Undo
	Redo
	Export

> **Tip**
>
> To view flow outcome data gathered using the **Flow Analytics** connector, click **Custom Analytics**.



The data in the **Custom Analytics Results** section is populated using custom flow analytics connectors.

The **Outcome Analytics** tab lets you view data about a selected flow's outcomes. The **Outcomes Over Time** chart shows the outcome numbers for each day and **Outcome Distribution** chart shows the relative occurrence of each outcome during the selected period of time.

The **Flow Analytics** tab lets you view data about a selected outcome in multiple flows. The **Flows Over Time** chart shows the number of times the outcome occurred in each flow for each day, and the **Flow Distribution** chart shows the relative occurrence of each flow reaching the outcome during the selected period of time.

On both tabs, you can use the **Date Range**, **Flow Filter**, and **Outcomes** selectors to adjust the content displayed in the charts. Click **Back** to return to the main dashboard.

Configuration Statistics

Configuration Statistics displays an overview of the number of active connectors, flows, and applications currently configured.

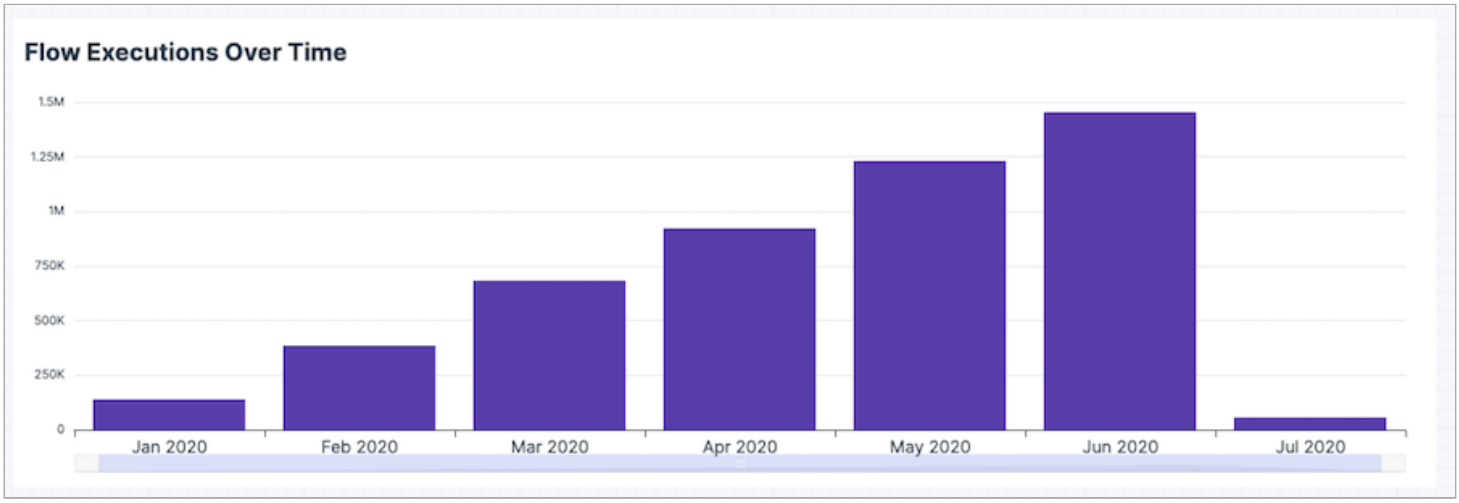
Click the **Connectors**, **Flows**, or **Applications** sections to view those configurations in a new tab.

Flow Executions

Flow Executions displays the total number of flow executions for the selected time period.

Flow Executions Over Time

Flow Executions Over Time displays the number of flow executions per day for the selected time period. Hover over each bar to view more specific data points.



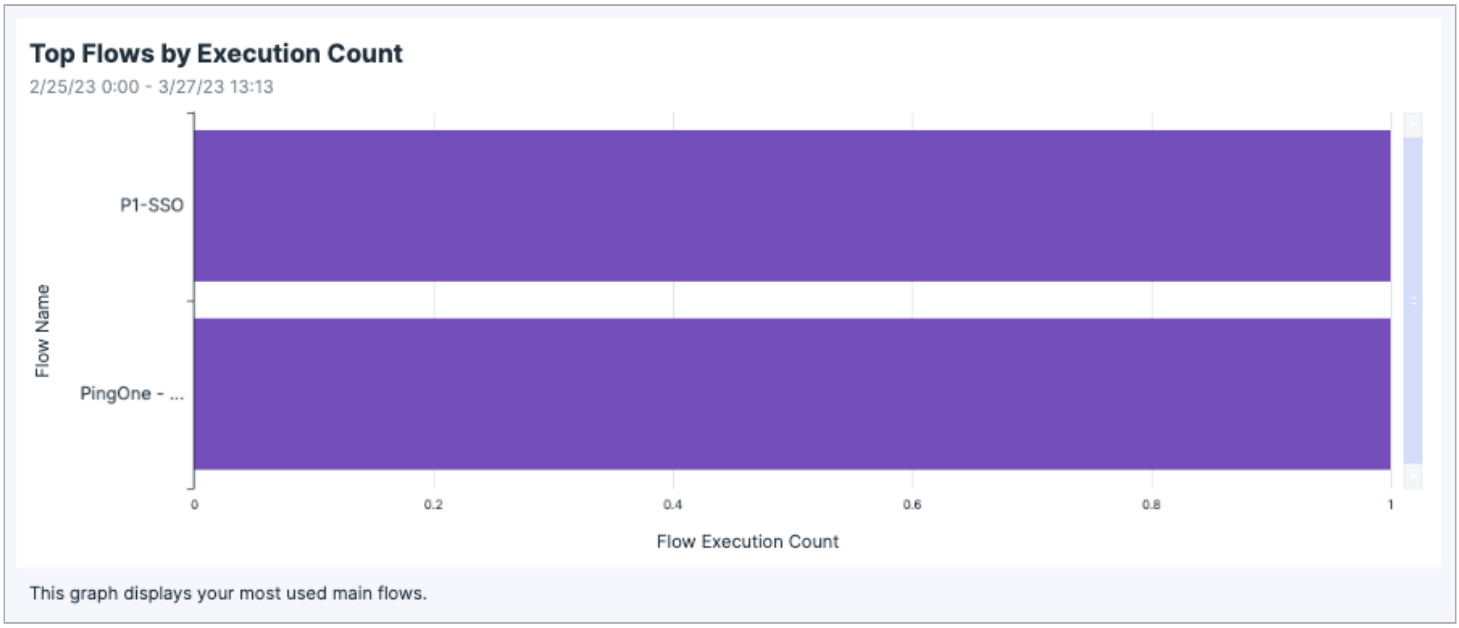
You can click and drag the ends of the horizontal slider bar to adjust the graph's range.

You can use the following icons to manipulate the data in the graph.

Icon	Description
	Maximize
	Drill up to the top from the resolved aggregation period.
	Drill up from the resolved aggregation period.
	Drill down from the resolved aggregation period.
	More options.

Top Flows by Execution Count

Top Flows by Execution Count displays your most used flows, determined by how many times a particular flow was used compared to other flows. Click and drag the ends of the vertical slider to adjust the graph's range.



The default data metrics are grouped by flow name and list each flow's execution count. Hover over a data point in the graph to view more specific information for that point.

Click the **Sort** ( ) icon to change the order of the graph bars.

Connector Executions

Connector Executions displays the total number of connector executions for the selected time period.

Connector Executions Over Time

Connector Executions Over Time displays the number of connector executions per day for the selected time period. Hover over each bar to view more specific data points.

You can click and drag the ends of the horizontal slider bar to adjust the graph's range.

You can use the following icons to manipulate the data in the graph.

Icon	Description
	Maximize
	Drill up to the top from the resolved aggregation period.
	Drill up from the resolved aggregation period.
	Drill down from the resolved aggregation period.

Icon	Description
	More options.

Top Connectors By Execution Count

Top Connectors by Execution Count displays your most used connectors, determined by how many times a particular connector was used compared to other connectors. Click and drag the ends of the vertical slider to adjust the graph's range.

The default data metrics are grouped by connector name and list each connector's execution count. Hover over a data point in the graph to view more specific information for that point.

Click the **Sort** () icon to change the order of the graph bars.

SDK Flow Executions Over Time

SDK Flow Executions Over Time displays the number of flow executions run using the SDK for the selected time period. Hover over each bar to view more specific data points.

You can click and drag the ends of the horizontal slider bar to adjust the graph's range.

Flow Executions by Integration

Flow Executions by Integration displays a pie chart of the execution methods used in your environment. Hover over each section to view more specific data points.

Custom Analytics

Custom Analytics displays flow results from the **Flow Analytics** connector.

The **Flow Analytics** connector lets you log a particular outcome within a flow, such as a login success or an OTP failure. You can use multiple instances of this connector to log the possible outcomes of your flows.

Outcome Analytics

The **Outcome Analytics** tab lets you view data about a selected flow's outcomes over a specified period of time. You can select the following parameters:

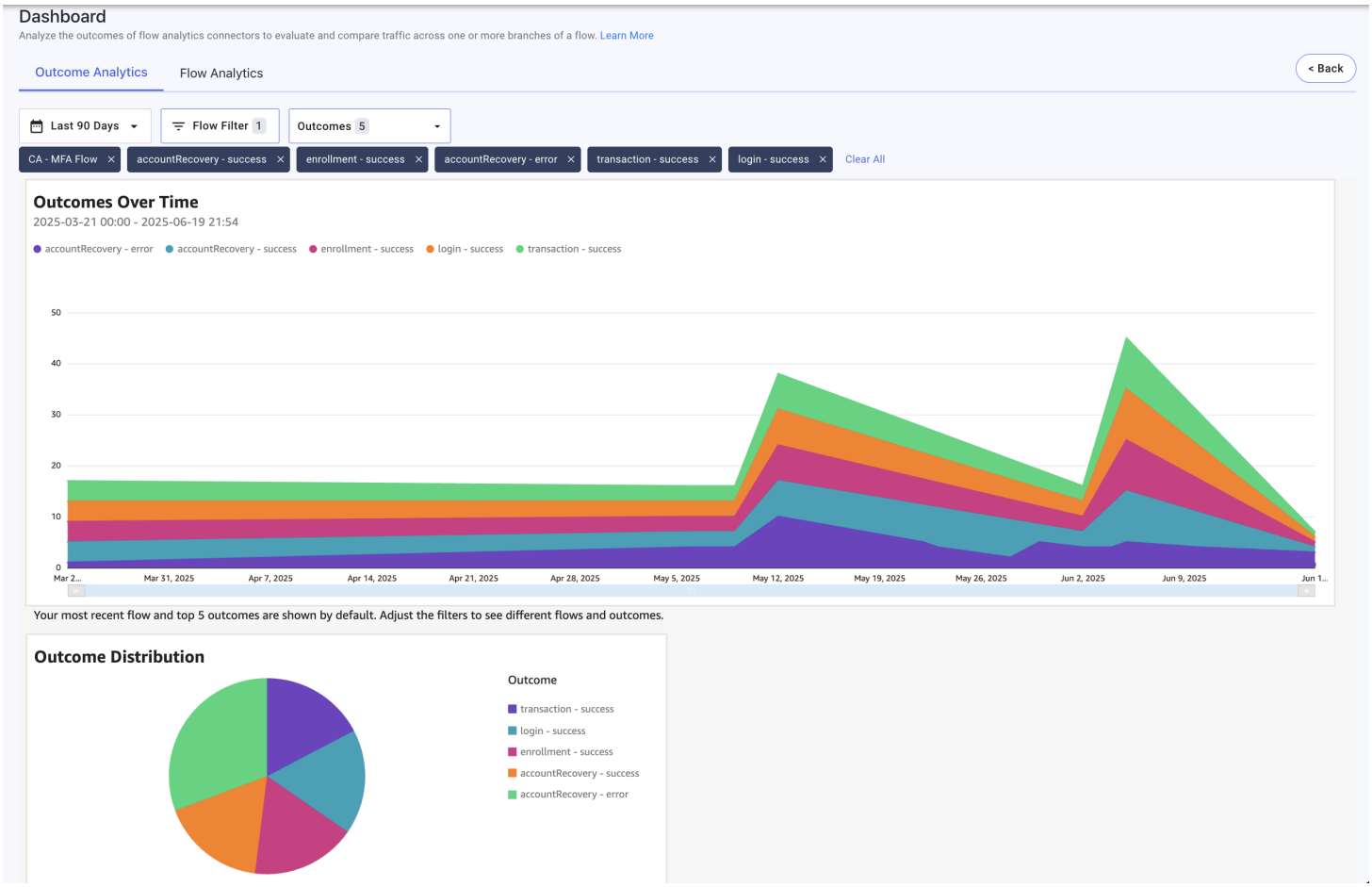
- A period of time within the past 180 days. The default period of time is **Today**.
- A flow with at least one **Flow Analytics** connector. The default flow is the first valid flow in the flow list.
- Up to five outcomes defined by **Flow Analytics** connector instances. The default outcomes are the five most common outcomes today.

The occurrences of each outcome are then displayed on two graphs:

- **Outcomes Over Time**: A line graph graph showing each outcome over the selected time period.

- **Outcome Distribution**: A pie chart showing the relative occurrence of each outcome.

You can hover over each graph to see the outcome counts for a specific date or outcome.



Flow Analytics

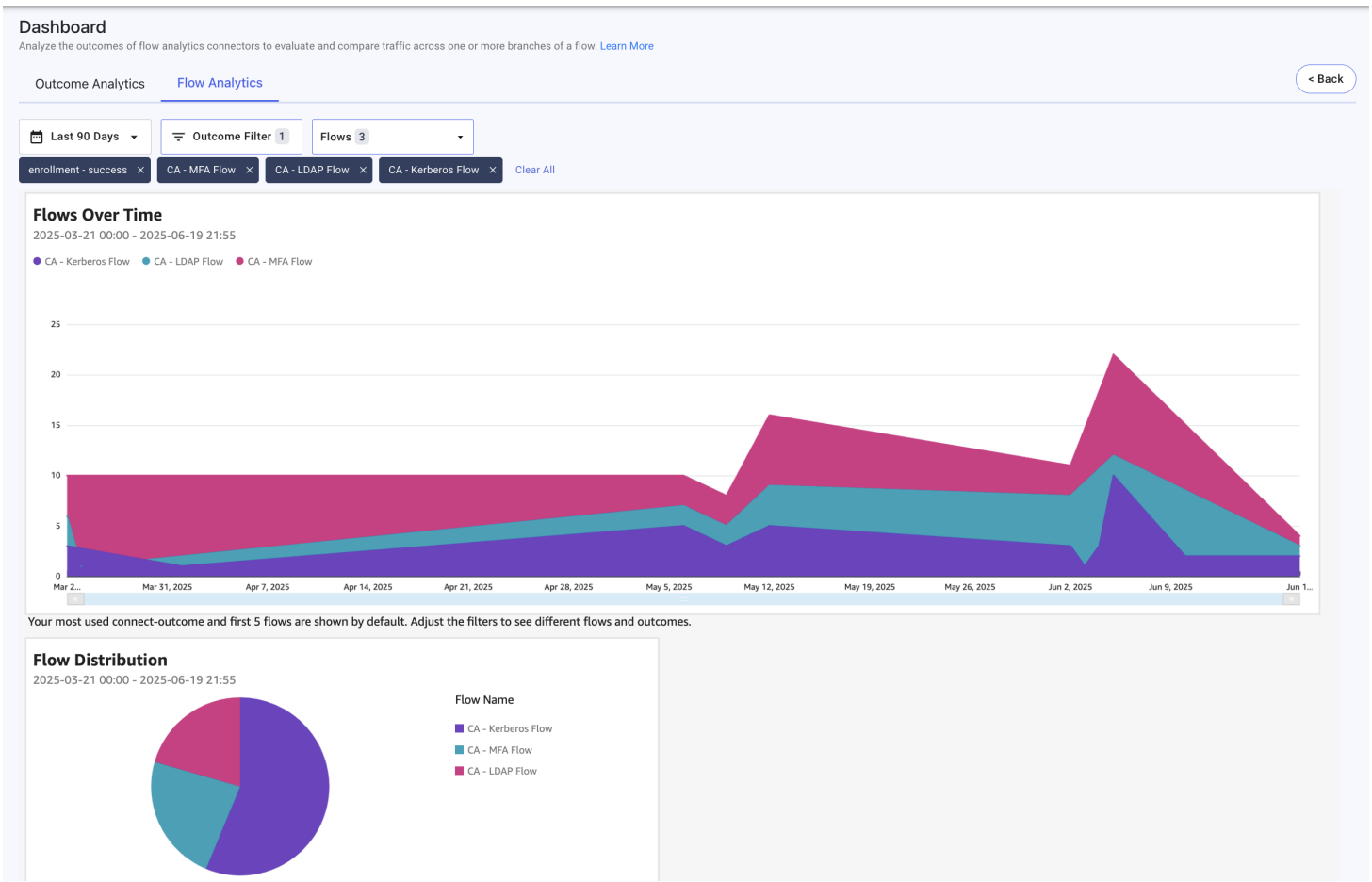
The **Flow Analytics** tab lets you view data about a selected outcome in multiple flows over a specified period of time. You can select the following parameters:

- A period of time within the past 180 days. The default period of time is **Today**.
- An outcome identified by the **Flow Analytics** connector. The default outcome is the most common outcome.
- Up to five flows that contain the specified outcome. The default flows are the first five in the list of available flows.

The occurrences of each outcome are then displayed on two graphs:

- **Flows Over Time**: A line graph graph showing the occurrence of the outcome in each flow over the selected time period.
- **Flow Distribution**: A pie chart showing the relative occurrence of the outcome in each flow.

You can hover over each graph to see the outcome counts for a specific date or outcome.



Current dashboard functionality notes

Review the following known issues and limitations regarding the current release of the PingOne DaVinci dashboard:

- When making a chart full screen, the axis labels might not appear. Use your browser's zoom in and out functionality (Ctrl + and Alt +, or Ctrl - and Alt -) to have them redrawn correctly.
- The **Drill up** and **Drill down** features are currently disabled.
- PingOne DaVinci does not support having the console open to different environments in separate browser tabs. The environment context can be mismatched when the user manually refreshes the page.
- **Top Flows** on the **Custom Analytics** dashboard is only shown for customers who were active before March 9, 2023. This is so that they can see their historical data. Newer customers don't see this chart, and the **Top Flows** on the main dashboard contains all of their data.

Company Settings

You can view and modify environment-wide settings in DaVinci.

Viewing company information

View your company ID and JSON Web Key Set.

The **Company** tab displays the following parameters:

- **Company ID**: This is a unique identifier for your current environment. It is used for launching flows and for configuring PingOne to work with DaVinci.
- **JWKS**: This section displays the environment's JSON Web Key Set. It is used for launching flows.

Changing the environment name

You can change the environment name that is displayed in the DaVinci user interface.

Steps

1. Click the **Company** tab.
2. In the **Name** field, enter a name for your environment.
3. Click **Save**.

Configuring a custom domain

Configure a domain for your company's DaVinci portal through PingOne.

Steps

1. Sign on to your PingOne environment.
2. Create a custom domain according to the Creating a Custom Domain in PingOne documentation.
3. If you are using the DaVinci external IDP in PingOne, update the URLs to use the new custom domain.
    1. Sign on to PingOne and go to **Integrations → External IDPs**.
    2. Open the PingOne DaVinci identity provider, then click the **Pencil** icon.
    3. Click the **Connection** tab.
    4. Update the **Authorization Endpoint**, **Token Endpoint**, **JWKS Endpoint**, and **Issuer** fields to use the custom domain.
    5. Click **Save**.

Managing debug logging

Enable or disable debug-level logging in your environment.

About this task

Since debug logging can include sensitive information, its use may not be appropriate in your environment. When enabled, debug logging can be activated for any flow in its flow settings. When disabled, debug logging cannot be activated in any flow.

Steps

1. Click the **Company** tab.
2. Select or deselect **Enable Analytics Debug View**.
3. Click **Save**.

Adding DaVinci Admin Users in PingOne

You can add admin users to DaVinci through PingOne.

Before you begin

To grant an admin role to a user, you must have an equal or greater role. To grant the **DaVinci Admin** role, you must have the **DaVinci Admin** role. To grant the **DaVinci Admin Read Only** role, you must have the **DaVinci Admin** or **DaVinci Admin Read Only** role. For more information about managing administrator roles in PingOne, see Managing administrator roles.

About this task

To add DaVinci admin users in PingOne:

Steps

1. Sign on to PingOne.
2. Select the environment that includes your administrative users.
3. Go to **Directory → Users**.
4. Locate the user that you want to edit. You can browse or search for users.
5. Click the **Details** icon to expand the user that you want to edit, and then click the **Pencil** icon.
6. Click **Roles**.
7. Add a role for the user.
    1. Click **+Add Role**.
    2. Select the **DaVinci Admin** role to grant the user full admin access, or select the **DaVinci Admin Read Only** role to grant the user read-only access.
    3. Click **Next**.
    4. In the **Define Responsibilities** section, click the **Add** icon to add the environment that includes your DaVinci instance.
    5. Click **Add Role**.

Result

The user can now access DaVinci through SSO.

Users

Users are end users created during flows. You can view and edit users' properties and delete users.

The **Users** table displays the following user properties.

Property	Description
User ID	The user ID.
Connection Name	The name of the connection which was used to create the user.
Full Name	The user's full given name.
Email Address	The user's email address.
Created Date	The date on which the user was first created.
User Name	The user's username.

You can view the following properties when editing a user.

Property	Description
Username	The user's username.
User Alias	If a team member has created an alias for the user, it is displayed.
User ID	The user ID.
Connection ID	The ID of the connection used to create the user.
Company ID	The ID of the company with which the user is associated.
Name	If the user's given name is known, it is displayed.
Email	If the user's email address is known, it is displayed.
Created Date	The date on which the user was first created.
Variables	If your environment includes one or more variables with the user context, the values of those variables are displayed.
Authentication Methods	If the user has configured one or more authentication methods, they are displayed.

Viewing and editing a user

View and edit a user's properties, including variables and user events.

Steps

1. Click the **Users** tab.
2. Find the user and click **Edit**.

> **Tip**
> Use the search pane if your environment has a large number of users.

3. **Optional:** To view user details, click **User Info**.
4. **Optional:** Add a user alias:
    1. Click **User Info**.
    2. Click **Add User Alias**.
    3. In the **User Alias** field, enter an internal alias for the user.
    4. Click **Update**.
5. **Optional:** To view a list of logged events involving the user, click **Events**.
6. Click **Close**.

Deleting a user

Delete a user to remove all information about the user from DaVinci.

Steps

1. Click the **Users** tab.
2. Find the user and click **Delete**.

> **Tip**
> Use the search pane if your environment has a large number of users.

   A confirmation message displays.

3. Click **Delete**.

Audit Trail

The **Audit Trail** tab provides a list of administrative events for the current user.

When you click the **Audit Trail** tab, the 10 most recent administrative events appear. Administrative events include:

- Login events for the DaVinci user interface
- Multi-factor authentication (MFA) events for the DaVinci user interface
- The creation or deletion of a flow, connector, application, or other entity
- The addition of custom JavaScript to a flow during flow creation, import, cloning or modification
- The addition of custom JavaScript to a connector that can run external scripts

Each event displays relevant information including:

- The event message, which indicates the type of event and its success or failure
- The date and time of the event
- The name and ID of the team member who initiated the event

To view additional event details, expand the event. You can click the **Next Page** and **Previous Page** icons to view earlier or later events.

Configuring SIEM Streaming

Configure SIEM streaming to send DaVinci events to a webhook configured in PingOne.

About this task

After you configure the PingOne webhook, it receives DaVinci events. These events are not affected by the logging level in the flow. The events use the payload structure described below.

> **Note**
>
> This is not a real-time service. There can be a delay between when the DaVinci event occurs and when it appears in PingOne or in a third-party service.

Steps

1. Sign on to PingOne.
2. Create a new webhook as described in the PingOne documentation.

   In the **Event types** list, select the **DaVinci** event type.

SIEM streaming payload structure

SIEM streaming events use the payload structure and properties described here.

Event Types

These event types generate a payload.

Custom Analytics

This event is sent by a Flow Analytics connector node within a flow. This node can be configured to send information about the flow outcome.

Start Interaction

This event is sent when a flow execution starts. It can be used to count the number of flow invocations during a particular time period.

Receive Request

This event is sent when a connector receives an event. It contains information about the capability to be used and its required inputs.

> **Note**
>
> This event type is deprecated.

Send Response

This event is sent after a connector has successfully executed a capability. It contains output information from the capability that successfully ran.

> **Note**
>
> This event type is deprecated.

Send Error Response

This event is sent after a connector has failed to execute a capability. It contains information about the error.

Payload Structure

All events use this payload structure.

```
{
  "id" : "payload ID",
  "recordedAt" : "timestamp",
  "correlationId" : "correlation ID",
  "action" : {
    "type" : "DAVINCI_INTERACTION.interaction_type",
    "description" : "description"
  },
  "resources" : {
    "id" : "flow ID",
    "type" : "DAVINCI_INTERACTION",
    "name" : "flow name",
    "environment" : {
      "id" : "company ID"
    }
  },
  "result" : {
    "status" : "status"
  },
  "_embedded" : {
    "flowInteractionEvent" : {
        // Actual Davinci Event Payload
    }
  }
}
```

Table 1. Event Properties

Property	Description
`id`	A unique identifier for the event payload.
`recordedAt`	The UTC timestamp when the event was recorded.
`correlationId`	A correlation ID for the event.
`action.type`	The type of event. Valid values are `START_INTERACTION`, `RECEIVE_REQUEST`, `SEND_RESPONSE`, and `SEND_ERROR_RESPONSE`.
`action.description`	A description of the event type.

Table 2. Resource Properties

Property	Description
`id`	The flow ID.
`type`	The type of interaction. All events currently have a type of `DAVINCI_INTERACTION`.
`name`	The flow name.
`environment.id`	The company ID.

Table 3. Result Properties

Property	Description
`status`	The result status. Valid values are `SUCCESS` and `FAILURE`.
`description`	A description of the response. This property is only present for `SEND_RESPONSE` events.

Table 4. Embedded Properties

Property	Description
`eventMessage`	The event type. Valid values are `StartInteraction`, `Receive Request`, `Send Response`, and `Send Error Response`.
`companyId`	The company ID.
`interactionId`	A unique identifier for the flow execution.
`flowVersionId`	The flow version ID. If the version ID is not available, the value is `-1`.
`identity`	The service that generated the event.
`tsEms`	The UTC timestamp when the event completed in DaVinci.
`flowID`	The ID of the flow.
`flowName`	The name of the flow.
`ID`	The node ID. This parameter is not sent for `START_INTERACTION` events for AP flows. **Note** This property will be deprecated in a future release.

Property	Description
`nodeID`	The node ID. This parameter is not sent for `START_INTERACTION` events for AP flows.
`originalCapabilityName`	The name of the capability used by the node. This parameter is not sent for `START_INTERACTION` events for AP flows.
`succcess`	A boolean value that indicates whether the capability succeeded or failed. This parameter is not sent for `START_INTERACTION` events for AP flows.
`connectorId`	The ID of the connector. This parameter is not sent for `START_INTERACTION` events for AP flows.
`nodeTitle`	The title of the node. This parameter is not sent for `START_INTERACTION` events for AP flows.
`nodeDescription`	The description value for the node. This parameter is not sent for `START_INTERACTION` events for AP flows.
`outcomeType`	The outcome type sent by the flow analytics connector (for example, login or enrollment). This parameter is only sent for custom analytics events.
`outcomeStatus`	The outcome status sent by the flow analytics connector (for example, success, error, denied, fraud, or approved). This parameter is only sent for custom analytics events.
`connectionId`	The ID of the flow analytics connector used to send the event. This parameter is only sent for custom analytics events.

Table 5. Parent Flow Properties

Property	Description
`companyId`	The company ID of the parent flow.
`flowID`	The ID of the flow.
`flowVersionId`	The flow version ID. If the version ID is not available, the value is `-1`.
`ID`	The node ID in the parent flow that triggered the current flow.
`connectionId`	The ID of the flow conductor connector used to launch the subflow.
`connectorId`	The name of the connector that launched the subflow.

Property	Description
`capabilityName`	The name of the capability used to launch the subflow. Valid values are `startUiSubFlow` for subflows with a UI component and `startSubFlow` for subflows without a UI component.
`success`	Indicates whether the subflow node was successful
`respondToUser`	Indicates whether the node presented a UI component to the user.
`interactionId`	The interaction ID for the parent flow.

Custom Analytics Example


```
{
  "id" : "c452dcdf-a535-43f6-8cc3-a09ccb440e91",
  "recordedAt" : "2024-02-15T16:07:03.995Z",
  "correlationId" : "002ec717-c5a6-44ca-9d6f-ec9a86282fe7",
  "action" : {
    "type" : "DAVINCI_INTERACTION.CUSTOM_ANALYTICS",
    "description" : "Davinci Interaction Custom Analytics"
  },
  "resources" : [ {
    "type" : "DAVINCI_INTERACTION",
    "id" : "79f303c7-f2cf-ae6c-5ce2-21dc013c80c5",
    "name" : "PingOne Sign On Augmented with Custom Analytics",
    "environment" : {
      "id" : "9f835dca-fa37-489b-b835-6587ef71e5d8"
    }
  } ],
  "result" : {
    "status" : "SUCCESS"
  },
 "_embedded" : {
   "flowInteractionEvent" : {
     "eventMessage" : "Custom Analytics",
     "interactionId" : "002ec717-c5a6-44ca-9d6f-ec9a86282fe7",
     "flowVersionId" : 4,
     "connectorId" : "analyticsConnector",
     "originalCapabilityName" : "logOutcome",
     "flowName" : "PingOne Sign On Augmented with Custom Analytics",
     "outcomeType" : "enrollment",
     "usageTransactionType" : "COUNTED",
     "companyId" : "9f835dca-fa37-489b-b835-6587ef71e5d8",
     "identity" : "analyticsConnector",
     "success" : true,
     "outcomeStatus" : "success",
     "packetProtocol" : "action",
     "connectionId" : "b50fb6e57556c2b3535d152758902e90",
     "nodeId" : "ezmu5wo88g",
     "tsEms" : "2024-02-15T16:07:03.947Z",
     "flowId" : "79f303c7f2cfae6c5ce221dc013c80c5"
   }
  }
}
```

Start Interaction Example


```
{
  "id" : "c84142a4-3b71-422c-aa77-9296e88d2881",
  "recordedAt" : "2023-03-22T04:28:55.395Z",
  "action" : {
    "type" : "DAVINCI_INTERACTION.START_INTERACTION"
  },
  "resources" : [ {
    "id" : "8f43a71f-85b5-2501-09e7-cf4b705b1446",
    "type" : "DAVINCI_INTERACTION",
    "name" : "8f43a71f85b5250109e7cf4b705b1446",
    "environment" : {
      "id" : "b5bbc401-7a1f-4738-b589-b3ea05bc46e4"
    }
  } ],
  "result" : {
    "status" : "SUCCESS"
  },
  "_embedded" : {
    "flowInteractionEvent" : {
      "eventMessage" : "Start Interaction",
      "companyId" : "b5bbc401-7a1f-4738-b589-b3ea05bc46e4",
      "interactionId" : "00467d91-1bb0-4ff7-ac70-307f49c3dcd2",
      "tsInteractionId" : "1679459335395 + 00467d91-1bb0-4ff7-ac70-307f49c3dcd2",
      "flowVersionId" : -1,
      "capabilityName" : "add",
      "identity" : "api",
      "tsEms" : "2023-03-22T04:28:55.395Z",
      "flowId" : "8f43a71f85b5250109e7cf4b705b1446",
      "packetTimestamp" : 1679459335395
    }
  }
}
```

Receive Request Example


```
{
  "id" : "2f8748d2-5c6b-4323-ba22-3276a5d54b86",
  "recordedAt" : "2023-03-22T04:28:55.456Z",
  "action" : {
    "type" : "DAVINCI_INTERACTION.RECEIVE_REQUEST"
  },
  "resources" : [ {
    "id" : "8f43a71f-85b5-2501-09e7-cf4b705b1446",
    "type" : "DAVINCI_INTERACTION",
    "name" : "8f43a71f85b5250109e7cf4b705b1446",
    "environment" : {
      "id" : "b5bbc401-7a1f-4738-b589-b3ea05bc46e4"
    }
  } ],
  "result" : {
    "status" : "SUCCESS"
  },
  "_embedded" : {
    "flowInteractionEvent" : {
      "eventMessage" : "Receive Request",
      "packetTo" : "httpConnector",
      "interactionId" : "00467d91-1bb0-4ff7-ac70-307f49c3dcd2",
      "flowVersionId" : 21,
      "capabilityName" : "add",
      "connectorId" : "httpConnector",
      "originalCapabilityName" : "customHtmlMessage",
      "companyId" : "b5bbc401-7a1f-4738-b589-b3ea05bc46e4",
      "tsInteractionId" : "1679459335456 + 00467d91-1bb0-4ff7-ac70-307f49c3dcd2",
      "identity" : "httpConnector",
      "packetProtocol" : "action",
      "connectionId" : "867ed4363b2bc21c860085ad2baa817d",
      "id" : "nzeo7no4po",
      "tsEms" : "2023-03-22T04:28:55.456Z",
      "flowId" : "8f43a71f85b5250109e7cf4b705b1446",
      "packetTimestamp" : 1679459335456
    }
  }
}
```

Send Response Example


```
{
  "id" : "971b5fd8-1b59-4e77-8836-00ab9941ff67",
  "recordedAt" : "2023-03-22T04:28:55.512Z",
  "action" : {
    "type" : "DAVINCI_INTERACTION.SEND_RESPONSE"
  },
  "resources" : [ {
    "id" : "8f43a71f-85b5-2501-09e7-cf4b705b1446",
    "type" : "DAVINCI_INTERACTION",
    "name" : "8f43a71f85b5250109e7cf4b705b1446",
    "environment" : {
      "id" : "b5bbc401-7a1f-4738-b589-b3ea05bc46e4"
    }
  } ],
  "result" : {
    "status" : "SUCCESS",
    "description" : "Send Response successful for flowId: 8f43a71f85b5250109e7cf4b705b1446, connector: httpConnector
and capability: customHtmlMessage"
  },
  "_embedded" : {
    "flowInteractionEvent" : {
      "eventMessage" : "Send Response",
      "packetTo" : "httpConnector",
      "interactionId" : "00467d91-1bb0-4ff7-ac70-307f49c3dcd2",
      "flowVersionId" : 21,
      "capabilityName" : "add",
      "connectorId" : "httpConnector",
      "originalCapabilityName" : "customHtmlMessage",
      "companyId" : "b5bbc401-7a1f-4738-b589-b3ea05bc46e4",
      "tsInteractionId" : "1679459335512 + 00467d91-1bb0-4ff7-ac70-307f49c3dcd2",
      "identity" : "httpConnector",
      "success" : true,
      "packetProtocol" : "action",
      "connectionId" : "867ed4363b2bc21c860085ad2baa817d",
      "id" : "nzeo7no4po",
      "tsEms" : "2023-03-22T04:28:55.512Z",
      "flowId" : "8f43a71f85b5250109e7cf4b705b1446",
      "packetTimestamp" : 1679459335512
    }
  }
}
```

Send Error Response Example


```
{
  "id" : "ccb8d1ec-61ed-4big-a0c4-a190ca677ad1",
  "recordedAt" : "2023-03-22T09:18:45.448Z",
  "action" : {
    "type" : "DAVINCI_INTERACTION.SEND_ERROR_RESPONSE"
  },
  "resources" : [ {
    "id" : "8f43a71f-85b5-2501-09e7-cf4b705b1446",
    "type" : "DAVINCI_INTERACTION",
    "name" : "8f43a71f85b5250109e7cf4b705b1446",
    "environment" : {
      "id" : "b5bbc401-7a1f-4738-b589-b3ea05bc46e4"
    }
  } ],
  "result" : {
    "status" : "SUCCESS"
  },
  "_embedded" : {
    "flowInteractionEvent" : {
      "eventMessage" : "Send Error Response",
      "packetTo" : "httpConnector",
      "interactionId" : "00152de6-e4db-4bbb-9bc7-1d09dc50492c",
      "flowVersionId" : 24,
      "capabilityName" : "add",
      "connectorId" : "httpConnector",
      "originalCapabilityName" : "makeRestApiCall",
      "companyId" : "b5bbc401-7a1f-4738-b589-b3ea05bc46e4",
      "tsInteractionId" : "1679476725448 + 00152de6-e4db-4bbb-9bc7-1d09dc50492c",
      "identity" : "httpConnector",
      "packetProtocol" : "action",
      "connectionId" : "867ed4363b2bc21c860085ad2baa817d",
      "id" : "yg52xyeh81",
      "tsEms" : "2023-03-22T09:18:45.448Z",
      "flowId" : "8f43a71f85b5250109e7cf4b705b1446",
      "packetTimestamp" : 1679476725448
    }
  }
}
```