Auth node reference

July 4, 2025

AM

Version: 7.5

Table of Contents

Basic nodes

Data Store Decision node

The **Data Store Decision** node checks that the credentials provided during authentication match the ones stored in the configured data store for the realm.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node requires the `realm`, `username`, and `password` properties in the incoming node state.

You can implement the following nodes as inputs to the Data Store Decision node:

Input nodes

This node requires the `username` and `password` properties in the incoming node state. Implement the following nodes earlier in the journey:

- Standalone PingAM deployment
  - Username Collector node
  - Password Collector node
- Ping Identity Platform deployment:
  - Platform Username node
  - Platform Password node

Alternatively, implement a Zero Page Login Collector node.

Dependencies

The Data Store Decision node is a basic node used in many types of authentication application types, such as basic, push, OAuth 2.0, and social provider authentication applications.

Configuration

This node has no configurable properties.

Outputs

This node copies shared and transient state into the outgoing node state.

Outcomes

Returns a boolean outcome:

`True`

The credentials match those found in the data store.

`False`

The credentials do *not* match those found in the data store.

Errors

The following Data Store Decision node warnings and errors can appear in the logs:

Warnings

- "invalid password error"
- "invalid username error"

Errors

- "Exception in data store decision node"

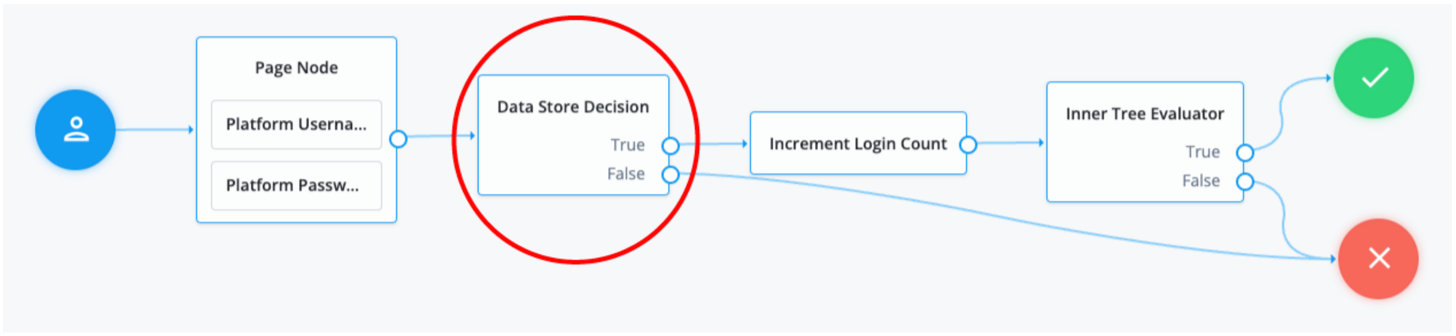
Troubleshooting

Review any errors and warnings this node logged.

- If this node logged a warning, fix the credentials and try again.
- If this node logged an error, review the log messages for the transaction to find the reason for the exception.

Examples

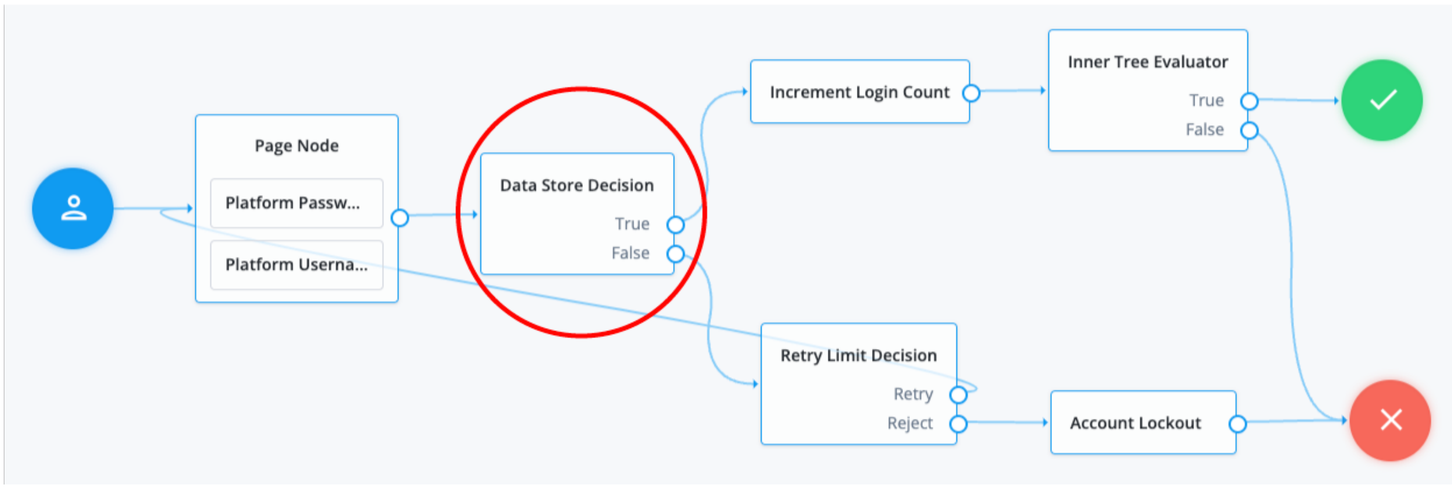
Example 1: Simple username and password collector nodes with Data Store Decision node



This example illustrates a simple login process. The journey involves a Page node that contains two embedded nodes: Platform Username node and Platform Password node. To enhance user experience, the Page node lets users input their username and password on a single page, instead of splitting them across two different pages.

The Data Store Decision node has two outcomes: `True` or `False`. When the outcome is `True`, it triggers a Login Count Decision node. The Increment Login Count node then moves to an Inner Tree Evaluator node, which performs additional login processes. The `False` outcome connects directly to a failure node, indicating a failed state where the username and/or password provided by the user did not match the information stored in the data store.

Example 2: Grant the user several attempts to enter their credentials correctly



In the following example, when an authentication attempt fails at the Data Store Decision node, you can direct it to a Retry Limit Decision node. The Retry Limit Decision node determines the number of retries allowed and either retries the login attempt or rejects it. If the journey rejects the login attempt after reaching the configured limit, for example three attempts, the operation results in an account lockout.

Alternate nodes

The LDAP Decision node supports LDAP Behera Password Policies with separate outcomes for accounts that are locked and passwords that have expired.

Failure node

The **Failure** node is a required element indicating the journey ended in failure.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

The failure outcomes of any preceding nodes.

Dependencies

None.

Configuration

This node has no configurable properties.

Outputs

None. The authentication journey ends in failure.

Outcomes

The authentication journey completes, ending in failure.

AM redirects the user to a failure URL.

Errors

The error depends on the **Authentication > Settings > Account Lockout > Login Failure Lockout Mode** setting for the realm (under **Native Consoles > Access Management**).

Without the setting enabled, by default, the node returns an error with a message such as the following:

```
{"code":401,"reason":"Unauthorized","message":"Login failure"}
```

With the setting enabled, the node checks the invalid attempts property of the user profile and does the following:

- Returns a warning message if the number of failed attempts is equal to or greater than the **Authentication > Settings > Account Lockout > Warn User After N Failures** setting:

```
{
  "code": 401,
  "reason": "Unauthorized",
  "message": "Warning: You will be locked out after 1 more failure(s).",
  "detail": {
    "failureUrl": ""
  }
}
```

- Increments the failure count in the user profile.

- Returns an error message if the account is `Inactive`:

```
{
  "code": 401,
  "reason": "Unauthorized",
  "message": "User Locked Out.",
  "detail": {
    "failureUrl": ""
  }
}
```

To troubleshoot an authentication failure, review the steps in the journey to find what caused the failure.

Examples

All authentication journeys have a **Failure** node as one of their terminals.

Kerberos node

Enables desktop single sign-on such that a user who has already authenticated with a Kerberos Key Distribution Center can authenticate to AM without having to provide the login information again.

To achieve this, the user presents a Kerberos token to AM through the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) protocol.

End users may need to set up Integrated Windows Authentication in Internet Explorer or Microsoft Edge to benefit from single sign-on when logged on to a Windows desktop.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `True`
- `False`

Evaluation continues along the `True` path if Windows Desktop SSO is successful; otherwise, evaluation continues along the `False` path.

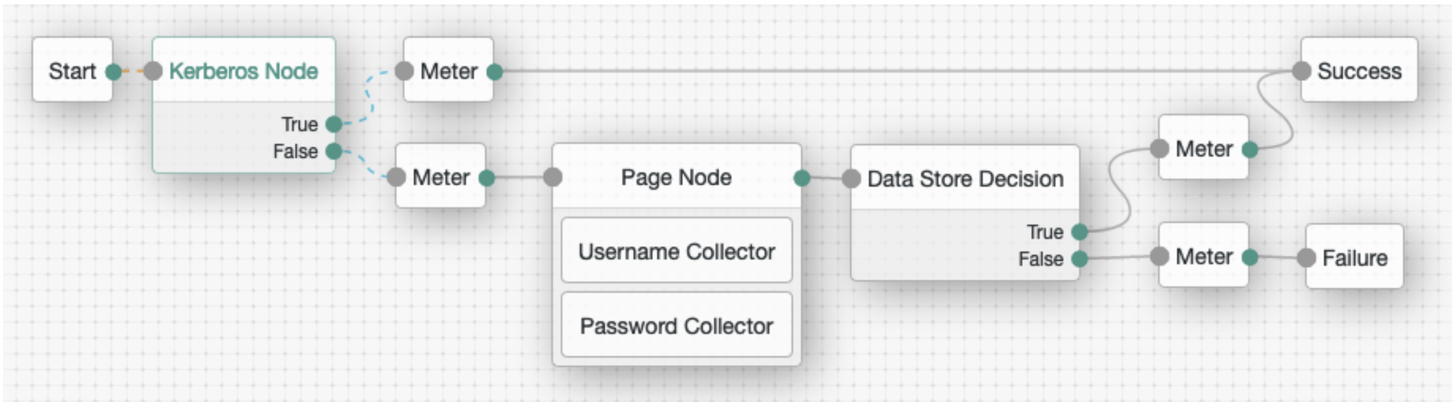
Properties

Property	Usage
Service Principal	Specifies the Kerberos principal for authentication in the format `HTTP/AM-DOMAIN@AD-DOMAIN`, where *AM-DOMAIN* corresponds to the host and domain names of the AM instance, and *AD-DOMAIN* is the domain name of the Kerberos realm (the FQDN of the Active Directory domain). *AD-DOMAIN* can differ from the domain name for AM. In multi-instance AM deployments, configure *AM-DOMAIN* as the FQDN or IP address of the load balancer in front of the AM instances. For example, `HTTP/AM-LB.example.com@KERBEROSREALM.INTERNAL.COM`.
Key Tab File Path	Specifies the full, absolute path of the keytab file for the specified Service Principal. **Tip** You generate the keytab file using the Windows `ktpass` utility; for example: `C:\> ktpass -out fileName.keytab -princ HTTP/openam.example.com@AD_DOMAIN.COM -pass +rdnPass -maxPass 256 -mapuser amKerberos@frdpcloud.com -crypto AES256-SHA1 -ptype KRB5_NT_PRINCIPAL -kvno 0`
Kerberos Realm	Specifies the name of the Kerberos (Active Directory) realm used for authentication. Must be specified in ALL CAPS.

Property	Usage
Kerberos Server Name	Specifies the fully qualified domain name, or IP address of the Kerberos (Active Directory) server.
Trusted Kerberos realms	Specifies a list of trusted Kerberos realms for user Kerberos tickets. If realms are configured, then Kerberos tickets are only accepted if the realm part of the user principal name of the user's Kerberos ticket matches a realm from the list. Each trusted Kerberos realm must be specified in all caps.
Return Principal with Domain Name	When enabled, AM returns the fully qualified name of the authenticated user rather than just the username.
Lookup User In Realm	Validates the user against the configured data stores. If the user from the Kerberos token is not found, evaluation continues along the `False` path. This search uses the `Alias Search Attribute Name` from the core realm attributes. Find more information about this property in [User profile.
Is Initiator	When enabled (`true`), specifies that the node is using *initiator* credentials, which is the default. When disabled (`false`), specifies that the node is using *acceptor* credentials.

Example

This flow attempts to authenticate the user with Windows Desktop SSO. If unsuccessful, AM requests the username and password for login. Meter nodes are used to track metrics for the various paths through the flow:



LDAP Decision node

The **LDAP Decision** node verifies that the provided username and password exist in the specified LDAP user data store. The node also checks whether the associated user account has expired or is locked out.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node requires the `username` and `password` properties in the incoming node state. Implement the following nodes earlier in the journey:

- Standalone PingAM deployment
  - Username Collector node
  - Password Collector node
- Ping Identity Platform deployment:
  - Platform Username node
  - Platform Password node

Alternatively, use the Zero Page Login Collector node.

Prerequisites

None

Configuration

Property	Usage
Primary LDAP Server *(required)*	Specify one or more primary directory servers. Specify each directory server in the following format: `host:port`. For example, `directory_services.example.com:389`.
Secondary LDAP Server	Specify one or more secondary directory servers. Specify each directory server in the following format: `host:port`. The journey uses the secondary servers when none of the primary servers are available. For example, `directory_services_backup.example.com:389`.

Property	Usage
DN to Start User Search *(required)*	Specify the DN from which to start the user search. More specific DNs, such as `ou=sales,dc=example,dc=com`, result in better search performance. If multiple entries with the same attribute values exist in the directory server, make sure this property is specific enough to return only one entry.
Bind User DN, **Bind User Password**	The credentials used to connect to the LDAP user data store.
Attribute Used to Retrieve User Profile *(required)*	The attribute used to retrieve a user profile from the directory server. The user search will have already happened, as specified by the **Attributes Used to Search for a User to be Authenticated** and **User Search Filter** properties.
Attributes Used to Search for a User to be Authenticated *(required)*	The attributes the node uses to match the credentials provided by the user to an entry in the directory server. For example, a value of `uid` forms the search filter `uid=user`. If you specify multiple values, such as `uid` and `cn`, the node forms a complex search filter `(|(uid=user)(cn=user))`. Multiple attribute values let the user authenticate with any one of the values. For example, if you set both `uid` and `mail`, then Barbara Jensen can authenticate with either `bjensen` or `bjensen@example.com`. **Note** If you're using account lockout and you set multiple attribute values here, you must add those attributes to the `Alias Search Attribute Name` property in the User profile.
User Search Filter	A filter to append to user searches. For example, if your search attribute is `mail` and you set **User Search Filter** to `(objectClass=inetOrgPerson)`, the node uses `(&(mail=address)(objectClass=inetOrgPerson))` as the resulting search filter. In this example, *address* is the mail address provided by the user.
Search Scope	The extent of the search for users in the directory server: • `OBJECT`: The search extends only to the entry specified by the **DN to Start User Search**. • `ONELEVEL`: The search extends to the entries that are direct children of the **DN to Start User Search**. • `SUBTREE`: The search extends to the **DN to Start User Search** and every entry under it. Default: `SUBTREE`
LDAP Connection Mode	Specifies whether to use SSL or StartTLS to connect to the directory server. The node must be able to trust the certificates used. Possible values: `LDAP`, `LDAPS`, and `StartTLS` Default: `LDAP`

Property	Usage
mTLS Enabled	Enables mTLS (mutual TLS) between AM and the directory server. This setting applies to *all* configured LDAP servers; that is, AM uses mTLS to authenticate to all LDAP servers configured for this node. When mTLS is enabled, AM ignores the values for **Bind User DN** and **Bind User Password**. If you enable this property, you must: • Set the **LDAP Connection Mode** to `LDAPS` • Provide an **mTLS Secret Label Identifier**] Default: Disabled
	Identifier used to create a secret label for mapping to the mTLS certificate in the secret store. AM uses this identifier to create a specific secret label for this node. The secret label takes the form `am.authentication.nodes.ldap.decision.mtls.identifier.cert` , where `identifier` is the value of **mTLS Secret Label Identifier**. The identifier can only contain alphanumeric characters (`a-z` , `A-Z` , `0-9`) and periods (`.`). It can't start or end with a period. All LDAP servers configured for this node share the same secret label. For more security, you should rotate certificates periodically. When you rotate a certificate, update the corresponding mapping in the realm secret store configuration to reflect this label. When you rotate a certificate, AM closes any existing connections using the old certificate. A new connection is selected from the connection pool and no server restart is required.
Return User DN to DataStore	When enabled, the node returns the DN rather than the User ID. From the DN value, AM uses the RDN to search for the user profile. For example, if a returned DN value is `uid=demo,ou=people,dc=openam,dc=example,dc=org` , AM uses `uid=demo` to search the directory server. Default: Enabled
User Creation Attributes	This list lets you map (external) attribute names from the LDAP directory server to (internal) attribute names used by AM.
Minimum Password Length	The minimum acceptable password length. Default: `8`
LDAP Behera Password Policy Support	When enabled, support interoperability with servers that implement the Internet-Draft, Password Policy for LDAP Directories. Default: Enabled
Trust All Server Certificates	When enabled, the server blindly trusts server certificates, including self-signed test certificates. Default: Disabled

Property	Usage
LDAP Connection Heartbeat Interval	Specifies how often AM should send a heartbeat request to the directory server to ensure that the connection doesn't remain idle. Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. Set the units for the interval in the `LDAP Connection Heartbeat Time Unit` property. **Note** Setting this property to `0` does *not* disable the heartbeat (keepalive) or load balancer availability checks. Disabling these features can only be configured at the global level. Default: `10`
LDAP Connection Heartbeat Time Unit	The time unit for the `LDAP Connection Heartbeat Interval`. Default: `seconds`
LDAP Operations Timeout	The timeout, in seconds, that AM should wait for a response from the directory server. Default: `0` (means no timeout)
Use mixed case for password change messages	Specifies whether the server returns password change messages in mixed (sentence) case or transforms them to uppercase. By default, the server transforms password reset and password change messages to uppercase. Enable this setting to return messages in sentence case. Default: Disabled
LDAP Affinity Level	Level of affinity used to balance requests across LDAP servers. Affinity-based load balancing means that each request for the same user entry goes to the same DS server. The DS server used for a specific operation is determined by the DN of the identity involved. List the directory server instances that form part of the affinity deployment in the **Primary LDAP Server** and **Secondary LDAP Server** properties. Options are: • `NONE` – no affinity • `BIND` – affinity for BIND requests only • `ALL` – affinity for all requests Default: `NONE`

Outcomes

`True`

The provided credentials match those found in the LDAP user data store.

`False`

The provided credentials don't match those found in the LDAP user data store.

`Locked`

The profile associated with the provided credentials is locked.

`Cancelled`

The user must change their password. When the journey prompts the user to change their password, the user cancels the password change.

`Expired`

The profile is found, but the password has expired.

Important

The LDAP Decision node *requires* specific user attributes in the LDAP user data store. These required attributes are present by default in PingDS. If you are using an alternative identity store, you might need to modify your LDAP schema to use this node.

Password Collector node

Prompts the user to enter their password.

The captured password is transient, persisting only until the authentication flow reaches the next node requiring user interaction.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	No

Outcomes

Single outcome path.

Evaluation continues after capturing the password.

Properties

This node has no configurable properties.

Success node

The **Success** node is a required element indicating the journey ended successfully.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

The success outcomes of any preceding nodes.

Dependencies

None.

Configuration

This node has no configurable properties.

Outputs

None.

Outcomes

The authentication journey completes successfully.

The node resets the failure count in the user profile when reached if the **User Status** property is set to `Active`.

Errors

- Checks the **Status** property of the user profile, when reached, and fails the authentication with an error message if the account is marked as `Inactive`:

```
{
    "code":401,
    "reason":"Unauthorized",
    "message":"User Locked Out.",
    "detail":
    {
        "failureUrl":""
    }
}
```

Examples

All authentication journeys have a **Success** node as one of their terminals.

Username Collector node

Prompts the user to enter their username.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	No

Outcomes

Single outcome path.

Evaluation continues after capturing the username.

Properties

This node has no configurable properties.

Zero Page Login Collector node

The **Zero Page Login Collector** node verifies the presence of specific HTTP username and password headers in the incoming authentication request. If the headers exist, the node uses their corresponding values as the provided username and password.

The **Zero Page Login Collector** node is commonly used to:

- Connect the `Has Credentials` outcome connector to the input of a Data Store Decision node.
- Connect the `No Credentials` outcome connector to the input of a Username Collector node followed by a Password Collector node (standalone AM) or a Platform Username node followed by a Platform Password node (Ping Identity Platform deployment), and then into the same Data Store Decision node. Find an example of this layout in the default `Example` authentication tree provided in AM.

The password collected by this node remains in the node state only until the journey reaches the next node that requires user interaction.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

- HTTP username header
- HTTP password header
- An allowlist of referrers if `Allow Without Referer` property is disabled. When you set the `Allow Without Referer` property to `false`, the request *must* contain a referrer from the allowlist; otherwise, the journey ends in a failure.

Dependencies

None.

Configuration

Properties

Property	Usage
Username Header name	Enter the name of the header that contains the username value. Default: **X-OpenAM-Username**
Password Header name	Enter the name of the header that contains the password value. Default: **X-OpenAM-Password**

Property	Usage
Allow without referer	If enabled, the node accepts incoming requests that do not contain a `Referer` HTTP header. If a `Referer` HTTP header is present, the value is not checked. If disabled, a `Referer` HTTP header must be present in the incoming request, and the value must appear in the Referer allowlist property. Default: `Enabled`
Referer Whitelist	Specify a list of URLs allowed in the `Referer` HTTP header of incoming requests. An incoming request containing a `Referer` HTTP header value not specified in the allowlist causes evaluation to continue along the `No Credentials` outcome path. **Note** You must disable the **Allow Without Referer** property for the referer allowlist property to take effect.

Outputs

The collected credentials from the headers.

Outcomes

- `Has Credentials`
- `No Credentials`

Evaluation continues along the `Has Credentials` outcome path if the specified headers are available in the request, or the `No Credentials` path if the specified headers are not present.

Errors

If more than one header value exists for username and/or password, the node returns the following error message

> "Expecting only one header value for username and/or password but size is {}."

If the node can’t decode the header values, the node returns the following error message

> "Could not decode username or password header."

Example

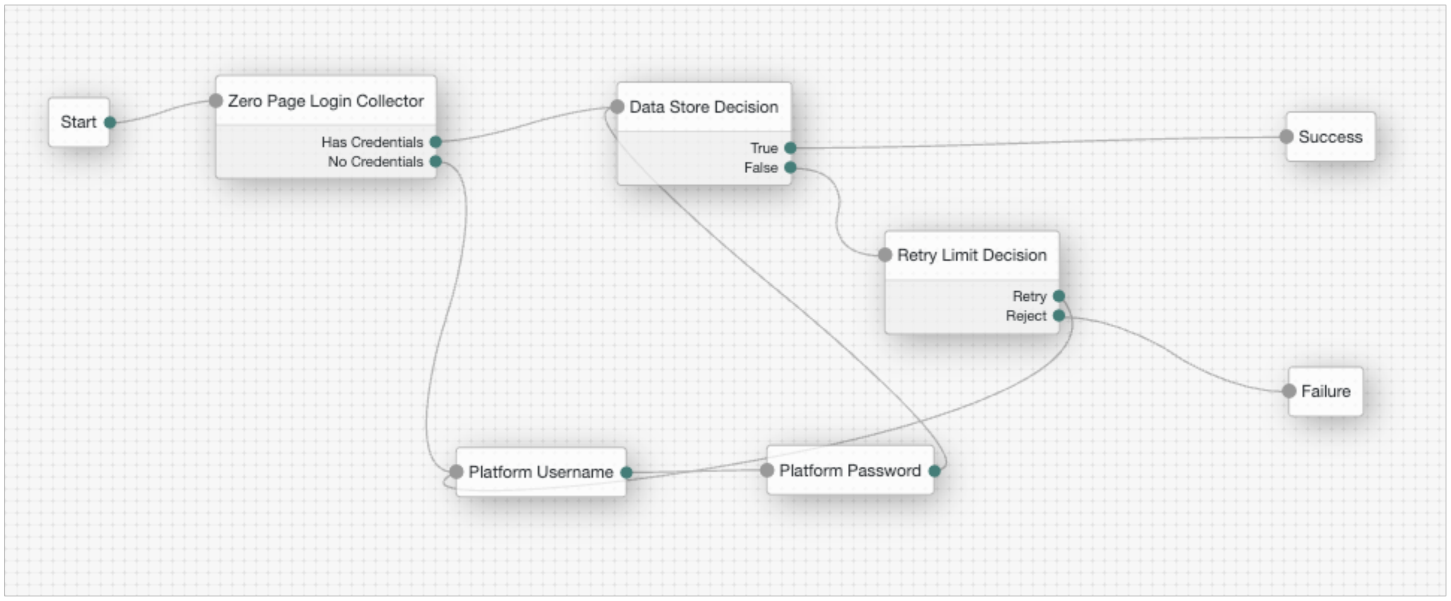


Figure 1. AM standalone

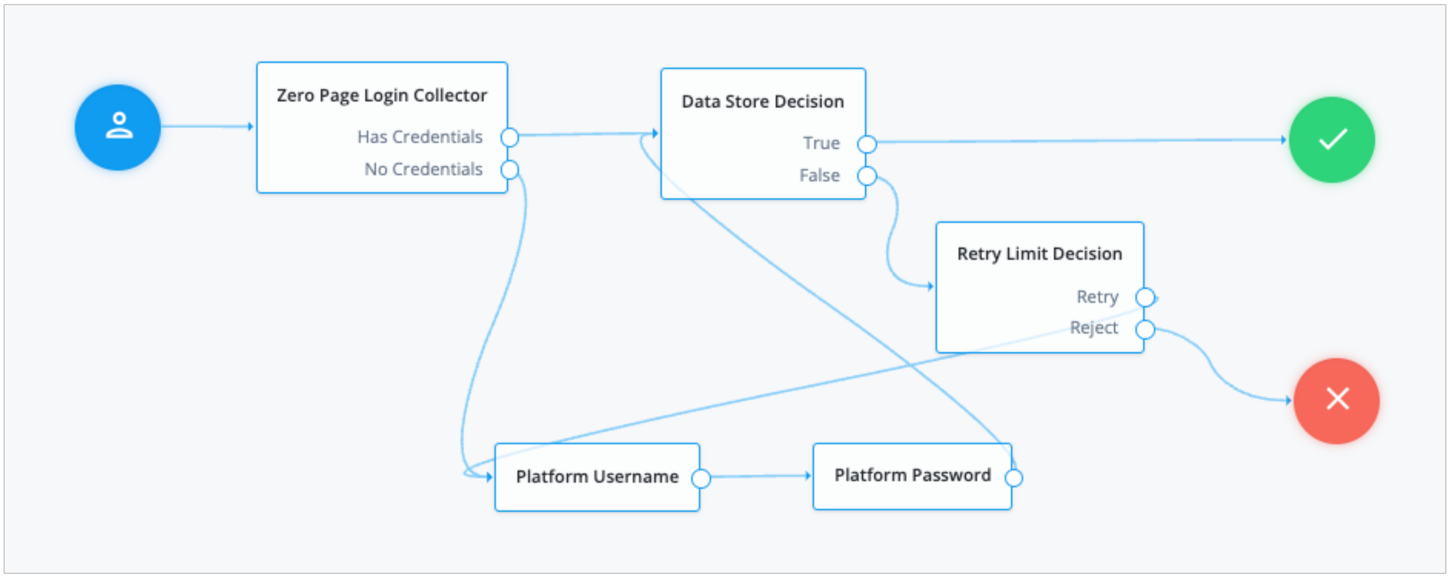


Figure 2. Ping Identity Platform deployment

Multi-factor nodes

Combined MFA Registration node

The **Combined MFA Registration node** lets an authenticated user register a device, such as a mobile phone, for multi-factor authentication with a push notification *and* an OATH one-time password in a single step.

This node can make journeys less complex by combining the functionality of the Push Registration node and OATH Registration node.

The node displays a single QR code that users scan to register their device for both push and OATH authentication. Journeys can then use the Push Sender node to verify possession of a registered device. If push does not succeed, for example, the user's device does not have internet access, the journey can fall back to using the OATH Token Verifier node to request a one-time passcode using OATH.

Learn more about push notifications and OATH one-time passwords in MFA: Push authentication and MFA: OATH authentication.]

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node requires a `username` in the incoming node state to identify which user is registering for MFA.

Implement a Username Collector node (standalone AM) or Platform Username node (Advanced Identity Cloud and Ping Identity Platform deployments) earlier in the journey.

Dependencies

You must configure the Push Notification service for the realm to use this node. Optionally, also configure the ForgeRock Authenticator (Push) service.

Find more information in the corresponding documentation for:

- Advanced Identity Cloud

- PingAM

Find information on provisioning the credentials used by the service in How To Configure Service Credentials (Push Auth, Docker) in Backstage.

Configuration

Property	Usage
Issuer	An identifier to appear on the user's device, such as a company name, a website, or a realm. The value is displayed by the authenticator application. For example, `Example Inc.` or the name of your application. Default: `ForgeRock`
Account Name	The profile attribute to display as the username in the authenticator application. If not specified, or if the specified profile attribute is empty, the username is used. Default: `Username`
Background Color	The background color in hex notation that displays behind the issuer's logo within the authenticator application. Default: `032b75`
Logo Image URL	The location of an image to download and display as the issuer's logo within the authenticator application. **Note** The ForgeRock Authenticator supports logos in JPEG and PNG format only. The application resizes your logo automatically, but a maximum image size of one MByte (or 1024 X 1024 pixels) is recommended. Default: none
Generate Recovery Codes	If enabled, recovery codes are generated and stored in the successful outcome's transient state. Use the Recovery Code Display node to display the codes to the user for safekeeping. Default: true **Important** Generating recovery codes overwrites all existing push-specific recovery codes. Only the most recent set of recovery codes can be used for authentication if a device has been lost or stolen.

Property	Usage
QR code message	A custom, localized message with instructions to scan the QR code to register the device. 1. Click **Add**. 2. Enter the message locale in the **Key** field; for example, `en-gb`. 3. Enter the message to display to the user in the **Value** field. Default: none
Registration Response Timeout	The period of time (in seconds) to wait for a response to the registration QR code. If no response is received during this time, evaluation continues along the `Time Out` outcome path. Default: `60`
One Time Password Length	The length of the generated OTP in digits. This value must be at least `6` and compatible with the hardware/software OTP generators you expect end users to use. For example, Google and ForgeRock authenticators support values of `6` and `8`, respectively. Default: `6`
Minimum Secret Key Length	Minimum number of hexadecimal characters allowed for the Secret Key. Default: `32`
OATH Algorithm	The algorithm the device uses to generate the OTP: ***HOTP*** HOTP uses a counter; the counter increments every time a new OTP is generated. When you use this setting, also set the same value in the OATH Token Verifier node. ***TOTP*** TOTP generates a new OTP every few seconds as specified by the `TOTP Time Step Interval` setting. Default: `TOTP`
TOTP Time Step Interval (`totpTimeInterval`)	The length of time that an OTP is valid in seconds. For example, if the time step interval is 30 seconds, a new OTP is generated every 30 seconds and is valid for 30 seconds only. Default: `30` seconds
TOTP Hash Algorithm	The HMAC hash algorithm used to generate the OTP codes. AM supports SHA1, SHA256, and SHA512. Default: `SHA1`
HOTP Checksum Digit	Add a digit to the end of the generated OTP to be used as a checksum to verify the OTP was generated correctly. This is in addition to the actual password length. Only set this if the user devices support it. Default: false

Property	Usage
HOTP Truncation Offset	An option used by the HOTP algorithm that not all devices support. Leave the default value unless you know user devices use an offset. Default: `-1`

JSON Authenticator Policies	Policies to apply to the device being registered, in JSON format. Use the following format to apply policies:
	`{ "[.var]#policyName#" : { "[.var]#policyParameters#" | [.label]#value# } }`
	Supported policies
	The ForgeRock Authenticator app supports enforcement of the following default policies:
	`biometricAvailable`
	Parameters: None
	The device must have a biometric sensor available and enabled in the operating system.
	`deviceTampering`
	Parameters: `score`
	The device must not have been tampered with; for example have root access or be jailbroken.
	This policy applies if the score returned by the device exceeds the provided `score` parameter, which is a number between `0` and `1.0`.
	Example:
	`{ "biometricAvailable": { }, "deviceTampering": { "score": 0.8 } }`

Outputs

- For Push registration, this node updates the shared state with the push device settings, the message ID, and the push challenge.
- For OATH registration, this node records the device profile in the `oathDeviceProfile` shared state attribute and the recovery codes in the `oathEnableRecoveryCode` shared state attribute.

Outcomes

Success

Device registration succeeded.

Failure

AM encountered an issue when attempting to register the authentication device.

Time Out

The node didn't receive a response from the device within the time specified in the configuration.

Errors

No username found

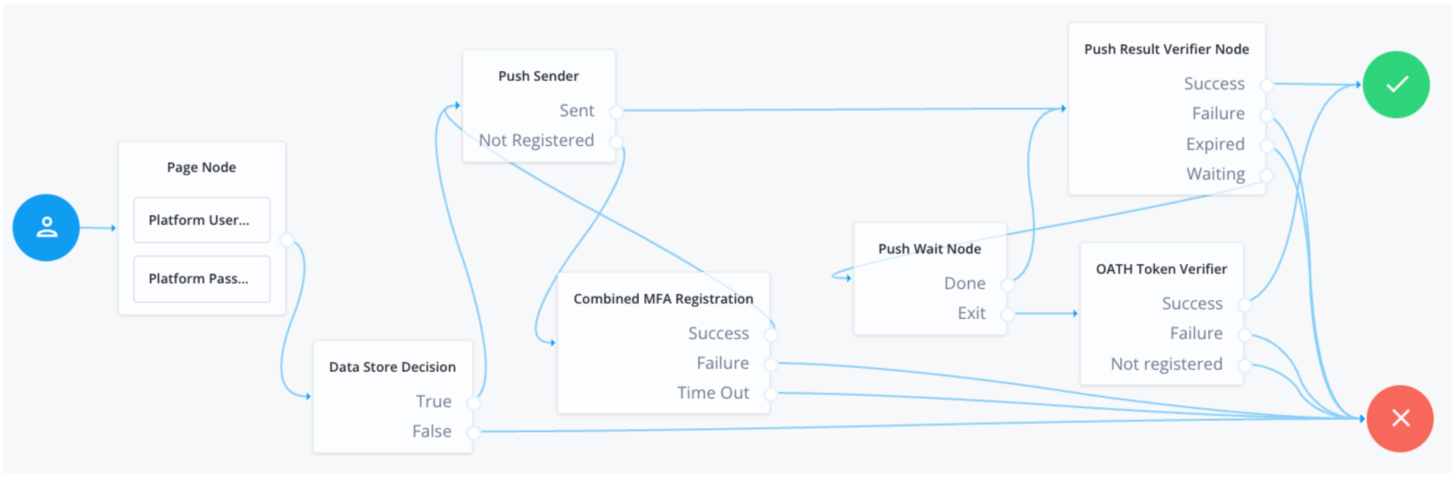
The node failed to read the username from the shared state.

Unable to find push message ID in sharedState

The node failed to read the push message ID from the shared state.

Examples

The following example shows an implementation of combined multi-factor registration in an authentication journey:



- The Page node with the Platform Username node and the Platform Password node prompts for the user credentials.
- The Data Store Decision node confirms the username-password credentials.
- The Push Sender node determines whether the user has a registered device.
  - If the user has a registered device:
    - The Push Sender node sends a push notification to the device.

    - The Push Result Verifier node validate the user's response to the push notification, looping through the Push Wait node until authentication succeeds.
    - The Push Wait node lets the user cancel the wait for a push notification. In this case evaluation continues to the OATH Token Verifier node, so the user can enter a one-time password instead.
  - If the user **doesn't** have a registered device:
    - The Push Sender node routes the user to the Combined MFA Registration node, which displays a QR code to the user to register a device.
    - After successful registration of a device for both push and OATH authentication, evaluation returns to the Push Sender node and continues with the registered device.

Device Binding node

Allows users to register one or more devices to their account. A user can bind multiple devices, and each device can be bound to multiple users.

You must ensure you authenticate the user and obtain their username in the journey before attempting to bind a device.

Registered devices share device data in the form of a public key and a key ID which AM stores in the user's profile, or you can save it in transient state for processing.

The private key of the keypair is kept safely on the device and secured with biometric security or a PIN.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Dependencies

You can verify possession of bound devices by using the Device Signing Verifier node.

You can save the device data to the user's profile by using the Device Binding Storage node.

To use Android Key Attestation, you must also configure the Android Key Attestation Service.

Configuration

Property	Usage
Authentication Type	Specifies how the device should secure access to the private key. The available options are: **`Biometric only`** Request that the client secures access to the cryptography keys with biometric security, such as a fingerprint. **`Biometric with PIN fallback`** Request that the client secures access to the cryptography keys with biometric security, such as a fingerprint, but allow use of a PIN if biometric is unavailable. **`Application PIN`** Request that the client secures access to the cryptography keys with an application-specific PIN. **Important** The PIN is not linked to the user's device PIN and is stored only on the client device. The PIN is not sent to AM, so if the user forgets their PIN, they must bind the device again. **`None`** Allow the client device to create the cryptography keys without securing access to them.
Application IDs	Specifies a list of Android package names and iOS bundle IDs of applications that are allowed to perform device binding. For example, `com.example.app`.
Title	Specifies a title to display to the user when asking them to bind the device.
Sub Title	Specifies a subtitle to display to the user when asking them to bind the device.
Description	Specifies descriptive text to display to the user when asking them to bind the device.
Maximum Saved Devices	Specifies the maximum number of devices stored in the user's profile. Set this property to `0` if you do not want to limit the number of devices. When this property is greater than zero, the `Exceed Device Limit` outcome path becomes available.
Timeout	Specify the number of seconds to wait for a response from the client during binding. If the specified time is reached, evaluation continues along the `Timeout` outcome path.

Property	Usage
Android Key Attestation	Use Android key attestation to increase confidence that the keys used by the bound device are valid, haven't been revoked, and use hardware-backed security storage. The attestation data is also stored in the transient state of the tree, in a variable named `DeviceBindingCallback.ATTESTATION`, so that you can access and parse the data in a scripted node if required. For information on the contents of the attestation data JSON response, refer to Attestation certificate in the Android documentation.
Store Device Data in Transient State	If enabled, the node *does not save* device data in the user's profile when it completes successfully. Instead, the node places the device information into the transient state in a variable named `DeviceBindingNode.DEVICE`. This allows subsequent nodes to use, parse, or alter the information before saving it. Use the Device Binding Storage node to save the device data to the user's profile. **Example device data**

```
{
  "uuid": "0ea44aa7-ef55-431b-885b-8c3a87e93331",
  "recoveryCodes": [],
  "deviceName": "Pixel 7 Pro",
  "deviceId":
"aaddfecd9b8b3e2a-153cae31c23bc51a8db6d71bc3a31423a6aca97d",
  "createdDate": 1694787036658,
  "lastAccessDate": 1694787036658,
  "key": {
    "kty": "RSA",
    "kid": "0ea44aa7-ef55-431b-885b-8c3a87e93331",
    "use": "sig",
    "alg": "RS512",
    "n": "n7nn76rmgcOGfuVm8N-wur4GgWW-
Iek0edwcQR865L3sjKON3XUCHi210tqMyc-PWlCaY-
dHisyy7TxK0jn4poui_aK3lnGYNzJpuyTU1-
sunSTRVMW8vDTEJxUNQMZFS086_8hVFiC9OnElkpFllp2jzfgZ7u318bdVMgib2bHlscyMo8
CZEwA_MHKteIkSD7CZIHMjm-
JlJIrKlaLIJ3lkZTUG29g2J9LvdGTMXyt206ZLQw3kAQ_QczHpiKieAiLd9sHydjB7BqGpgC
xjCkmqVi4BEvMl8sEEFnpZG1NzjrCBnGfSWr83dzenr6tbdCh5iew-BIdDXXaDPOXRew",
    "e": "AQAB"
  }
}
```

Android key attestation

When binding a device running Android N (`24`) or newer, you can use Android key attestation to increase confidence that the keys used by the bound device are valid, have not been revoked, and use hardware-backed security storage.

The ForgeRock SDK for Android generates attestation data for the cryptographic keys it uses for device binding. Using information provided by Google, including a certificate revocation status list (CRL) and hardware attestation root certificate, the node can verify that the certificates are trustworthy.

If you enable the **Android key attestation** property and the device is running an earlier Android version, evaluation continues down the `Unsupported` outcome path.

Android key attestation *is not* supported if you select `Application PIN` in the **Authentication Type** property. Evaluation continues down the `Unsupported` outcome path in this case.

The node does not attempt attestation when binding non-Android devices.

Outputs

If you enable the **Android Key Attestation** property, the node outputs attestation data in a variable named `DeviceBindingCallback.ATTESTATION`.

If you enable the **Store Device Data in Transient State** property, the node outputs device data in a variable named `DeviceBindingNode.DEVICE`.

Outcomes

- `Success`
- `Failure`
- `Exceed Device Limit`
- `Unsupported` (Client)
- `Abort` (Client)
- `Timeout` (Client)

If the user successfully binds their device, evaluation continues along the `Success` outcome path.

If AM encounters an issue when attempting to register using a device, evaluation continues along the `Failure` outcome path.

If the **Maximum Saved Devices** property is set to an integer greater than zero, and binding a new device would take the number of devices above the specified threshold, then evaluation continues down the `Exceed Device Limit` outcome path. In this case, you need to instruct your users to log in with an existing bound device in order to remove one or more of their registered devices.

If the user's client does not support the requested operation, evaluation continues along the `Unsupported` outcome path. For example, the node is configured to require biometric authentication, but the device does not provide support.

If the user cancels the attempt to bind a device, evaluation continues along the `Abort` outcome path.

If the node does not receive a response from the user's device within the **Timeout** specified in the node configuration, evaluation continues along the `Timeout` outcome path.

Device Binding Storage node

Persists collected device binding data to a user's profile in the identity store.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

The node uses the variable named `DeviceBindingNode.DEVICE` as input from the state.

The node requires that a Device Binding node has gathered device data previously in the journey.

Outcomes

`True`

Device binding data was successfully stored in the user's profile.

`False`

Device binding data was not stored in the user's profile.

Properties

None.

Device Signing Verifier node

Verifies possession of a registered bound device.

The node requires the device to sign a challenge string using the private key that corresponds to a stored public key.

The user might need to unlock their cryptography keys with biometric security — such as a fingerprint — or a PIN.

Note

This node can be used in usernameless authentication flows.
The ForgeRock SDKs store and provide the identity when handling the callbacks from this node. If the device has been registered by more than one user, the SDK displays a list of the registered keys to choose from on the client device.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

If you want the device to sign a particular challenge, the value must be available in shared state.

Dependencies

You can bind devices by using the Device Binding node.

Configuration

Property	Usage
Sign Random Challenge	Specifies the node should use a random value as the challenge for signing.
Shared state attribute for Challenge	Specifies the node should use a value from the named attribute in shared state as the challenge for signing.
Application IDs	Specifies a list of Android package names and iOS bundle IDs of applications that are allowed to perform device signing verification. For example, `com.example.app`.
Title	Specifies a title to display to the user when asking them to bind the device.
Sub Title	Specifies a secondary, or subtitle to display to the user when asking them to bind the device.
Description	Specifies descriptive text to display to the user when asking them to bind the device.
Timeout	Specify the number of seconds to wait for a response from the client during binding. If the specified time is reached, evaluation continues along the `Timeout` outcome path.

Outcomes

- `Success`
- `Failure`
- `No Registered Device`

- `Key Not Found`
- `Unsupported` (Client)
- `Abort` (Client)
- `Timeout` (Client)
- `ClientNotRegistered` (Client)

If the response from the device is verified as coming from a bound device, evaluation continues along the `Success` outcome path.

If AM cannot verify that the response was signed by a bound device, evaluation continues along the `Failure` outcome path.

If the user does not have any bound devices, evaluation continues along the `No Registered Device` outcome path. The user is determined either previously in the authentication journey, or by reading the `sub` claim from the response when doing usernameless flows.

If the client device cannot access the cryptography keys, or the key ID that AM requested cannot be located, evaluation continues along the relevant `Key Not Found` outcome path.

If the user's client does not support the requested operation, evaluation continues along the `Unsupported` outcome path.

If the user cancels authentication, evaluation continues along the `Abort` outcome path.

If the node does not receive a response from the user's device within the **Timeout** specified in the node configuration, evaluation continues along the `Timeout` outcome path.

If the client device does not have the keys present to be able to sign the challenge, evaluation continues along the `ClientNotRegistered` outcome path.

Get Authenticator App node

The **Get Authenticator App** node displays information to get an authenticator app from the Apple App Store or the Google Play Store.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

None. This node doesn't read shared state data.

Dependencies

This node has no dependencies.

Configuration

Property	Usage
Get App Authenticator Message	(Optional) Add a custom, localized message to display to the user. You can use the following variables to customize the message: • `{{appleLink}}` • `{{appleLabel}}` • `{{googleLink}}` • `{{googleLabel}}` You can also include HTML elements, for example: `Apple: <a target='_blank' href='{{appleLink}}'>{{appleLabel}}</a>` 1. Click **+**. 2. In the **Key** field, enter the locale. For example, `en-gb`.(1) 3. In the **Value** field, enter the message. 4. Click **Done**. 5. Repeat to add more messages and save your changes when you're done. Leave blank to use the default message.(2) Default: `Get the app from the {{appleLink}} or on {{googleLink}}.`
Continue Label	(Optional) Add custom, localized text to display on the button the user can click to continue. 1. Click **+**. 2. In the **Key** field, enter the locale. For example, `en-gb`.(1) 3. In the **Value** field, enter the message. 4. Click **Done**. 5. Repeat to add more messages and save your changes when you're done. Leave blank to use the default message.(2) Default: `Continue`
Apple App Store URL	The URL to download your authenticator app from the Apple App Store. The default value points to the ForgeRock Authenticator app for iOS. Default: `https://itunes.apple.com/app/forgerock-authenticator/id1038442926`
Google Play URL	The URL to download your authenticator app from the Google Play Store. The default value points to the ForgeRock Authenticator app for Android. Default: `https://play.google.com/store/apps/details?id=com.forgerock.authenticator`

(1) Specify a locale that Java supports, such as `en-gb`. Otherwise, the node throws a configuration exception with an `Invalid locale provided` message.

(2) PingAM only: Learn more about customizing and translating default messages in Internationalize nodes.

Outputs

This node doesn't change the shared state.

Outcomes

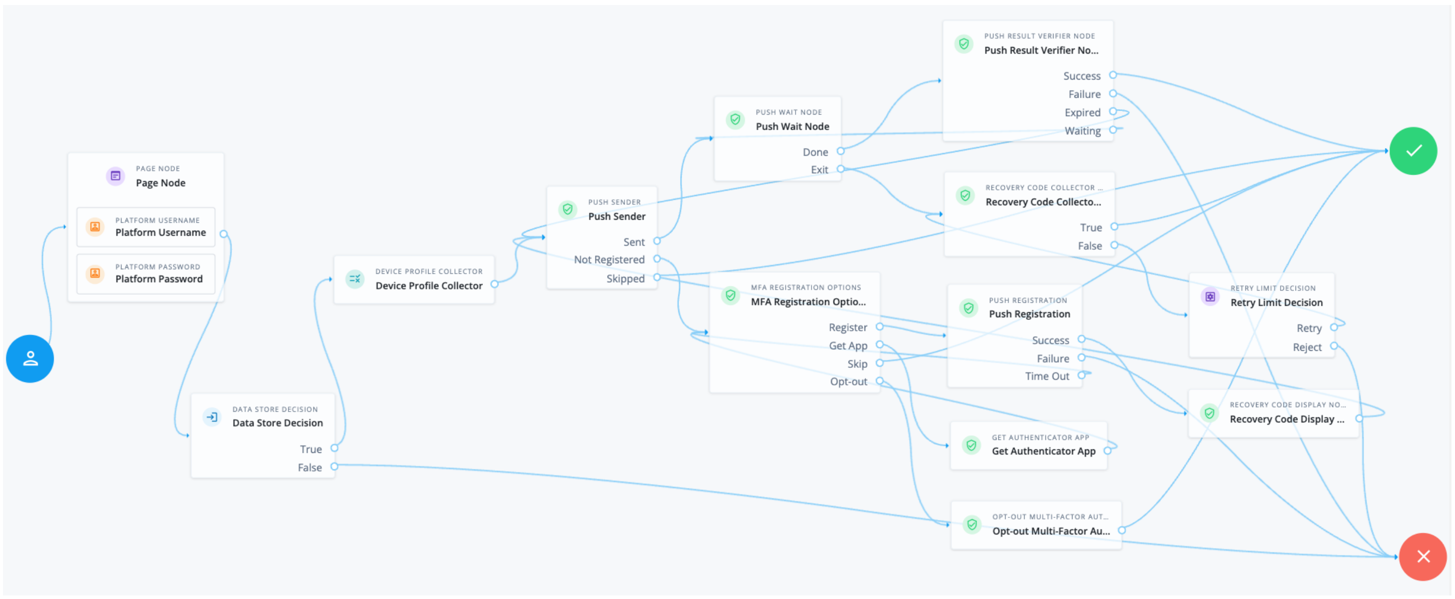
Single outcome path.

Errors

This node doesn't log any error or warning messages of its own.

Examples

The following example shows one possible implementation of multi-factor push authentication, which uses this node:



List of node connections

Source node	Outcome path	Target node
Page Node containing nodes to collect credentials. For standalone AM deployments, implement a Username Collector node and a Password Collector node. For Advanced Identity Cloud and Ping Identity Platform deployments, implement a Platform Username node and a Platform Password node.	→	Data Store Decision
Data Store Decision	True	Device Profile Collector
	False	Failure
Device Profile Collector	→	Push Sender
Push Sender	Sent	Push Wait
	Not Registered	MFA Registration Options
	Skipped	Success
Push Wait	Done	Push Result Verifier
	Exit	Recovery Code Collector Decision
Push Result Verifier	Success	Success
	Failure	Failure
	Expired	Push Sender
	Waiting	Push Wait
MFA Registration Options	Register	Push Registration
	Get App	Get Authenticator App
	Skip	Success
	Opt-out	Opt-out Multi-Factor Authentication
Recovery Code Collector Decision	True	Success
	False	Retry Limit Decision

Source node	Outcome path	Target node
Push Registration	Success	Recovery Code Display Node
	Failure	Failure
	Time Out	MFA Registration Options
Get Authenticator App	→	MFA Registration Options
Opt-out Multi-Factor Authentication	→	Success
Retry Limit Decision	Retry	Recovery Code Collector Decision
	Reject	Failure
Recovery Code Display Node	→	Push Sender

After verifying the user's credentials, evaluation continues to the Device Profile Collector node to collect the device's location and then proceeds to the Push Sender node.

If the user *has* a registered device:

1. The Push Sender node sends a push notification to their registered device.
2. The Push Wait node pauses authentication for five seconds. During this time, the user can respond to the push notification on their device using the ForgeRock Authenticator app.

   If the user exits the Push Wait node, they're directed to the Recovery Code Collector Decision node, where they can enter a recovery code to authenticate.

   > **Tip**
   >
   > Configure the **Exit Message** property in the Push Wait node with a message, such as `Lost phone? Use a recovery code` for situations like this.

   A Retry Limit Decision node allows three attempts to enter a recovery code before failing the authentication.
3. The Push Result Verifier node verifies the user's response:
   - If the user responds positively, they're authenticated successfully and logged in.
   - If the user responds negatively, authentication fails.
   - If the push notification expires, the Push Sender node sends a new push notification.

     > **Tip**
     >
     > Use a Retry Limit Decision node to constrain the number of times a new code is sent.
   - If the user hasn't yet responded, the flow loops back a step and the Push Wait node pauses authentication for another 5 seconds.

If the user *doesn't have* a registered device:

1. The MFA Registration Options node presents the user with the following options:

    Register Device

    The flow continues to the Push Registration node, which displays a QR code for the user to scan with their authenticator app.

    Get the App

    Displayed only if the node is configured to display Get Authenticator App. The flow continues to the Get Authenticator App node, which displays links to download the authenticator app.

    Skip this step

    Displayed only if the node is configured to allow users to skip registration. In this example, skipping is linked to the `Success` outcome. However, you could provide an alternative authentication flow using an Inner Tree Evaluator node for example.

    Opt-out

    Displayed only if the node is configured to allow users to skip registration. Evaluation continues to the Opt-out Multi-Factor Authentication node, which updates the user's profile to skip MFA with push in the future. In this example, after updating the profile, the flow continues to the `Success` outcome.

2. The user registers the device with the Push Registration node.

    After registration, the Recovery Code Display node displays the recovery codes to the user and the flow returns to the Push Sender node to continue push authentication.

> **Note**
>
> To manage push devices, the user must log in using either the device or a recovery code.
> Find more information in the MFA documentation for:
>
> - PingAM
> - Advanced Identity Cloud

HOTP Generator node

Creates a string of random digits of the specified length for use as a one-time password.

Passwords are stored in the `oneTimePassword` transient node state property.

Use this node with these nodes to add one-time password verification as an additional factor:

- OTP Email Sender node
- OTP SMS Sender node
- OTP Collector Decision node

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

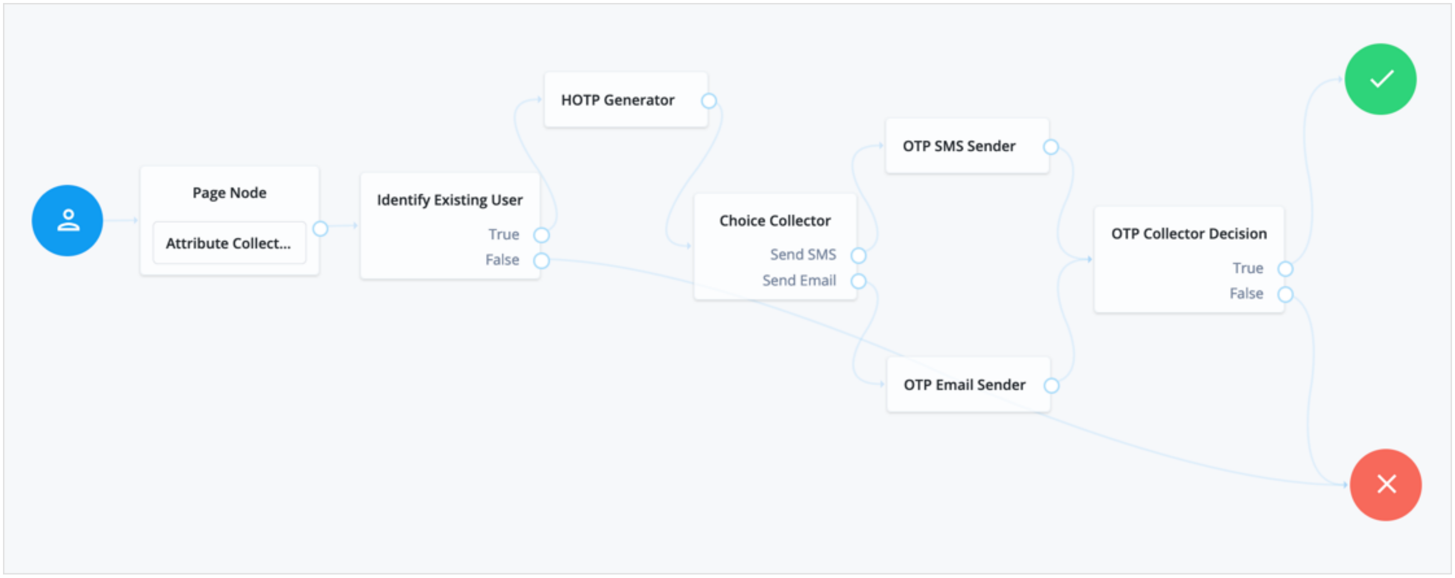
Single outcome path.

Properties

Property	Usage
One-time password length	Specify the number of digits in the one-time password. The minimum number of digits is 6, in accordance with the HOTP specification. Default: 8

Example

The following example uses an HOTP generator as part of multi-factor authentication:



MFA Registration Options node

The **MFA Registration Options** node lets the user register a multi-factor authentication (MFA) device or skip the registration process.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

- This node requires a `username` in the incoming node state to identify which user to update.

Implement a Username Collector node (standalone AM) or Platform Username node (Advanced Identity Cloud and Ping Identity Platform deployments) earlier in the journey.

- This node requires the `mfaMethod` in the incoming state to know what type of MFA device to register:
  - For push authentication, this node requires the `pushMessageId` in the incoming state, which is a unique ID to identify the push notification request.

    Implement a Push Sender node earlier in the journey.
  - For OATH authentication, implement the OATH Token Verifier node earlier in the journey.

Dependencies

This node has no dependencies.

Configuration

Property	Usage
Remove 'skip' option	Select this option to make it mandatory for the user to register a device. When selected, the **Skip this Step** and **Opt-out** buttons aren’t displayed. When disabled, the user can skip device registration or opt-out if required.
Display Get Authenticator App	Select this option to display the **Get the App** button.

Property	Usage
Message	(Optional) Add a custom, localized message to display to the user with instructions on interacting with this node: 1. Click **+**. 2. In the **Key** field, enter the locale. For example, `en-gb`.[1] 3. In the **Value** field, enter the message. 4. Click **Done**. 5. Repeat to add more messages and save your changes when you're done. Leave blank to use the default message.[2] Default: `On this page you can choose to register, skip or opt-out the second factor authentication method selected to protect your account. If you "Skip", an MFA method will not be registered now, but you will be prompted again on your next login. Otherwise, if you "Opt out", an MFA method will not be registered now and you will not be asked again. This choice is not recommended.`
Register Device	(Optional) Add custom, localized text to display on the button the user can click to register their device: 1. Click **+**. 2. In the **Key** field, enter the locale. For example, `en-gb`.[1] 3. In the **Value** field, enter the message. 4. Click **Done**. 5. Repeat to add more messages and save your changes when you're done. Leave blank to use the default message.[2] Default: `Register Device`
Get Authenticator App	(Optional) Add custom, localized text to display on the button the user can click to get the authenticator app: 1. Click **+**. 2. In the **Key** field, enter the locale. For example, `en-gb`.[1] 3. In the **Value** field, enter the message. 4. Click **Done**. 5. Repeat to add more messages and save your changes when you're done. Leave blank to use the default message.[2] Default: `Get the App`

Property	Usage
Skip this Step	(Optional) Add custom, localized text to display on the button the user can click to skip registering their device: 1. Click **+**. 2. In the **Key** field, enter the locale. For example, `en-gb`.[1] 3. In the **Value** field, enter the message. 4. Click **Done**. 5. Repeat to add more messages and save your changes when you're done. Leave blank to use the default message.[2] Default: `Skip this step`
Opt-out	(Optional) Add custom, localized text to display on the button the user can click to opt out of registering their device: 1. Click **+**. 2. In the **Key** field, enter the locale. For example, `en-gb`.[1] 3. In the **Value** field, enter the message. 4. Click **Done**. 5. Repeat to add more messages and save your changes when you're done. Leave blank to use the default message.[2] Default: `Opt-out` **Note** This node doesn't update the user's profile with the opt-out decision. Use the **Opt-out** outcome in an Opt-out Multi-Factor Authentication node to persist the decision in the user's profile.

[1] Specify a locale that Java supports, such as `en-gb`. Otherwise, the node throws a configuration exception with an `Invalid locale provided` message.

[2] PingAM only: Learn more about customizing and translating default messages in Internationalize nodes.

Outputs

This node doesn't change the shared state.

Outcomes

`Register`

The user chooses to register an MFA device.

`Get App`

The user chooses to get the authenticator app.

`Skip`

The user chooses to skip registering an MFA device this time.

`Opt-out`

The user chooses to opt-out of registering an MFA device.

Errors

The node can log the following errors:

- `Expected username to be set`

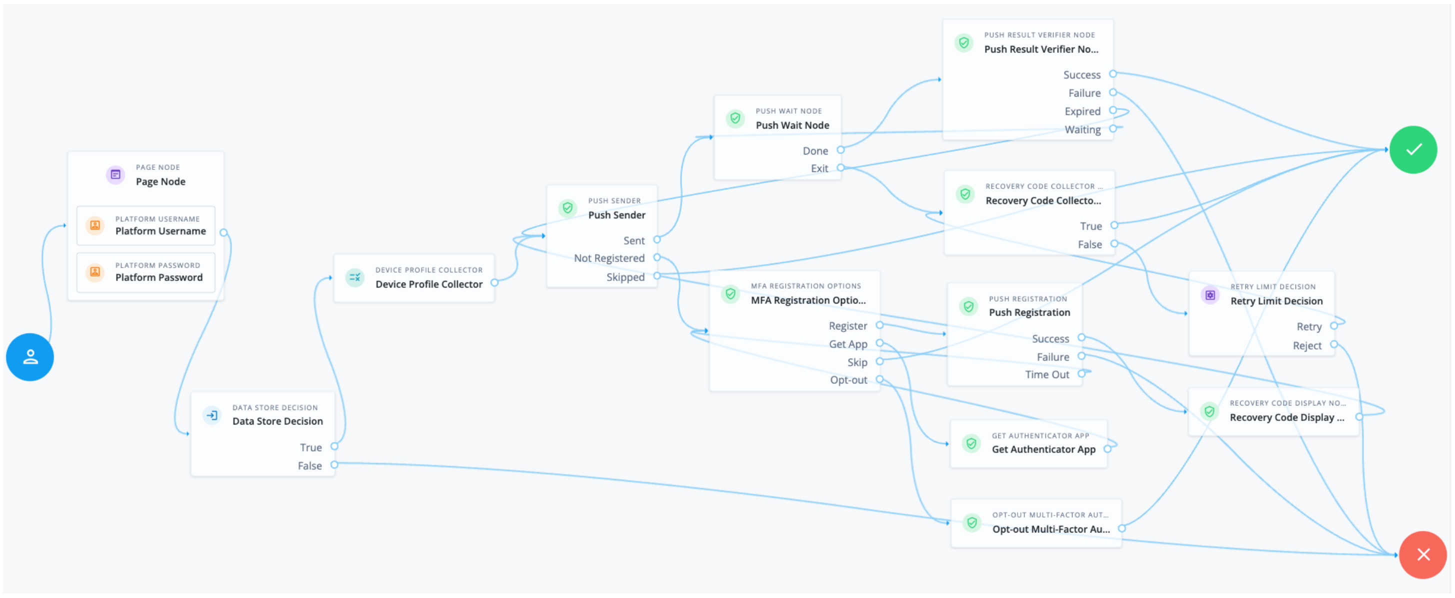
  The node can't identify the user from the shared state.

- `Expected multi-factor authentication method to be set`

  The node can't identify the MFA method from the shared state.

Examples

The following example shows one possible implementation of multi-factor push authentication, which uses this node:



List of node connections

Source node	Outcome path	Target node
Page Node containing nodes to collect credentials. For standalone AM deployments, implement a Username Collector node and a Password Collector node. For Advanced Identity Cloud and Ping Identity Platform deployments, implement a Platform Username node and a Platform Password node.	→	Data Store Decision
Data Store Decision	True	Device Profile Collector
	False	Failure
Device Profile Collector	→	Push Sender
Push Sender	Sent	Push Wait
	Not Registered	MFA Registration Options
	Skipped	Success
Push Wait	Done	Push Result Verifier
	Exit	Recovery Code Collector Decision
Push Result Verifier	Success	Success
	Failure	Failure
	Expired	Push Sender
	Waiting	Push Wait
MFA Registration Options	Register	Push Registration
	Get App	Get Authenticator App
	Skip	Success
	Opt-out	Opt-out Multi-Factor Authentication
Recovery Code Collector Decision	True	Success
	False	Retry Limit Decision

Source node	Outcome path	Target node
Push Registration	Success	Recovery Code Display Node
	Failure	Failure
	Time Out	MFA Registration Options
Get Authenticator App	→	MFA Registration Options
Opt-out Multi-Factor Authentication	→	Success
Retry Limit Decision	Retry	Recovery Code Collector Decision
	Reject	Failure
Recovery Code Display Node	→	Push Sender

After verifying the user's credentials, evaluation continues to the Device Profile Collector node to collect the device's location and then proceeds to the Push Sender node.

If the user *has* a registered device:

1. The Push Sender node sends a push notification to their registered device.
2. The Push Wait node pauses authentication for five seconds. During this time, the user can respond to the push notification on their device using the ForgeRock Authenticator app.

   If the user exits the Push Wait node, they're directed to the Recovery Code Collector Decision node, where they can enter a recovery code to authenticate.

   Tip

   Configure the **Exit Message** property in the Push Wait node with a message, such as `Lost phone? Use a recovery code` for situations like this.

   A Retry Limit Decision node allows three attempts to enter a recovery code before failing the authentication.
3. The Push Result Verifier node verifies the user's response:
   - If the user responds positively, they're authenticated successfully and logged in.
   - If the user responds negatively, authentication fails.
   - If the push notification expires, the Push Sender node sends a new push notification.

     Tip

     Use a Retry Limit Decision node to constrain the number of times a new code is sent.
   - If the user hasn't yet responded, the flow loops back a step and the Push Wait node pauses authentication for another 5 seconds.

If the user *doesn't have* a registered device:

1. The MFA Registration Options node presents the user with the following options:

   Register Device

   The flow continues to the Push Registration node, which displays a QR code for the user to scan with their authenticator app.

   Get the App

   Displayed only if the node is configured to display Get Authenticator App. The flow continues to the Get Authenticator App node, which displays links to download the authenticator app.

   Skip this step

   Displayed only if the node is configured to allow users to skip registration. In this example, skipping is linked to the `Success` outcome. However, you could provide an alternative authentication flow using an Inner Tree Evaluator node for example.

   Opt-out

   Displayed only if the node is configured to allow users to skip registration. Evaluation continues to the Opt-out Multi-Factor Authentication node, which updates the user's profile to skip MFA with push in the future. In this example, after updating the profile, the flow continues to the `Success` outcome.

2. The user registers the device with the Push Registration node.

   After registration, the Recovery Code Display node displays the recovery codes to the user and the flow returns to the Push Sender node to continue push authentication.

Note

To manage push devices, the user must log in using either the device or a recovery code.
Find more information in the MFA documentation for:

- PingAM
- Advanced Identity Cloud

OATH Device Storage node

The **OATH Device Storage** node stores devices in the user profile after an OATH Registration node records them in the shared state.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes

Product	Available?
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Authenticators

The OATH-related nodes can integrate with the following authenticator apps:

- The ForgeRock Authenticator app for Android and iOS.
- Third-party authenticator apps that support the following open standards:
    - RFC 4226: HMAC-Based One-Time Password (HOTP)
    - RFC 6238: Time-Based One-Time Password (TOTP)

Inputs

This node reads the device profile as the value of the shared state attribute `oathDeviceProfile`.

Dependencies

Precede this node in the flow with an OATH Registration node with its **Store device data in shared state** setting enabled.

Configuration

This node has no configurable properties.

Outputs

This node doesn't change the shared state.

Outcomes

`Success`

The node wrote the device profile to the user's account.

`Failure`

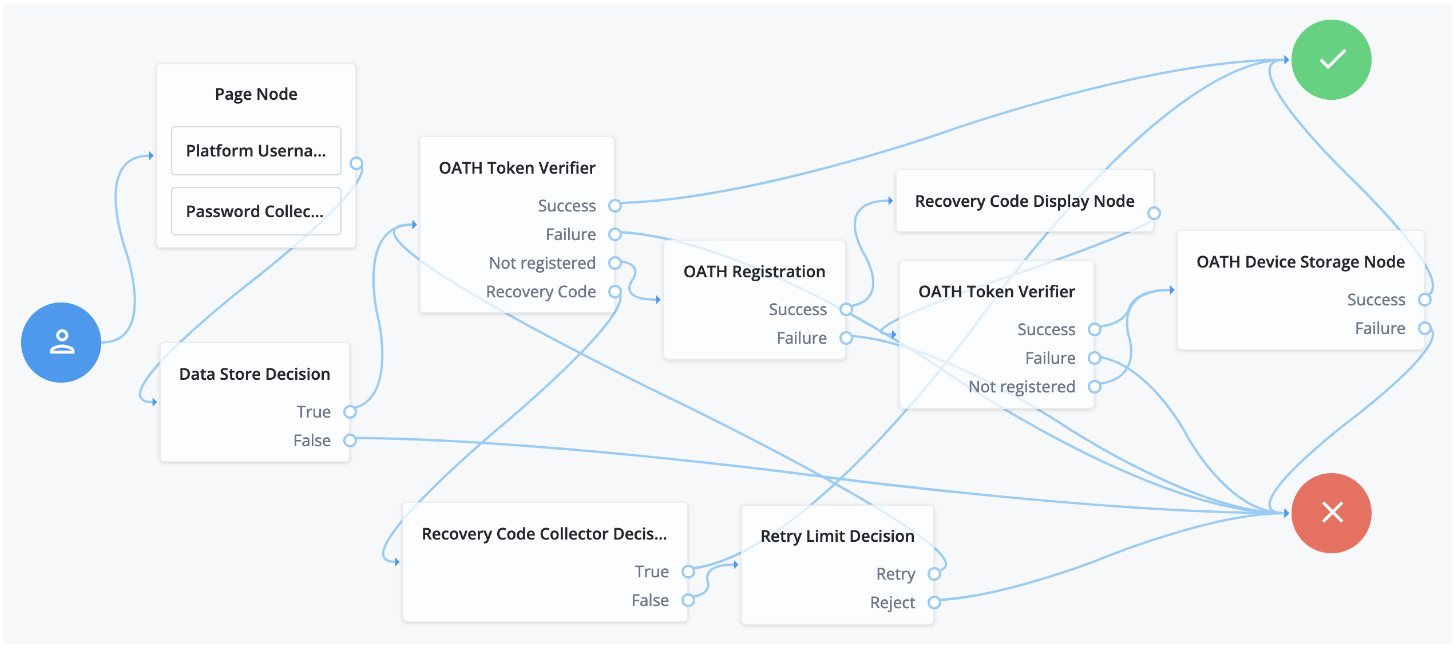
Any other case.

Errors

This node logs a `No device profile found on shared state` error message if it can't get the device profile from the `oathDeviceProfile` shared state attribute.

Example

The following journey includes both username-password and one-time passcode authentication:



- The Page node with the Platform Username node and the Platform Password node prompts for the user credentials.
- The Data Store Decision node confirms the username-password credentials.
- The first OATH Token Verifier node prompts for a one-time passcode with an option to use a recovery code.
- The OATH Registration node prompts the user to register a device and includes its profile in the shared state.
- The Recovery Code Display node shows the recovery codes and prompts the user to keep them safe.
- The second OATH Token Verifier node prompts for a one-time passcode using the newly registered device.
- The OATH Device Storage node writes the device profile to the user's account.
- The Recovery Code Collector Decision node prompts for a recovery code.
- The Retry Limit Decision node lets the user retry another code if they enter one incorrectly.

OATH Registration node

The **OATH Registration** node lets the user register a device for OATH-based multi-factor authentication (MFA).

Based on the node settings, the user device displays a QR code that includes all the details required for registration. If registration is successful, the node stores the device data, and recovery codes (if enabled), and sets the `skippable` attribute to prevent repeat registration at next login.

Tip

You can use the Combined MFA Registration node to register a device for both push notifications and one-time password (OATH) verification in a single step.
Refer to the OATH Token Verifier node example that demonstrates how use to use other MFA nodes to create a complete OATH authentication journey.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Authenticators

The OATH-related nodes can integrate with the following authenticator apps:

- The ForgeRock Authenticator app for Android and iOS.
- Third-party authenticator apps that support the following open standards:
  - RFC 4226: HMAC-Based One-Time Password (HOTP)
  - RFC 6238: Time-Based One-Time Password (TOTP)

Inputs

This node reads the `username` attribute and optionally the `oathDeviceProfile` attribute from the shared state.

Dependencies

Confirm the user credentials before letting them register a device. For example, precede this node with the following nodes earlier in the authentication flow:

- Username Collector node (standalone AM) or Platform Username node (Ping Identity Platform deployment)
- Password Collector node (standalone AM) or Platform Password node (Ping Identity Platform deployment)
- Data Store Decision node

Properties

Property	Usage
Issuer	Specify an identifier to appear on the user's device, such as a company name, a website, or a realm. The authenticator application displays the value. Default: `ForgeRock`
Account Name	Select the profile attribute to display as the username in the authenticator application. If not specified, or if the specified profile attribute is empty, their username is used. Default: **Username**
Background Color	The background color in hex notation that displays behind the issuer's logo within the authenticator application. Default: `032b75`
Logo Image URL	The location of an image to download and display as the issuer's logo within the authenticator application. **Note** The ForgeRock Authenticator supports logos in JPEG and PNG format only. The application resizes your logo automatically, but a maximum image size of one MByte (or 1024 X 1024 pixels) is recommended. Default: none
Generate Recovery Codes	If enabled, recovery codes are generated and stored in the successful outcome's transient state. Use the Recovery Code Display node to display the codes to the user for safekeeping. Default: true
QR code message	A custom, localized message with instructions to scan the QR code to register the device. 1. Click **Add**. 2. Enter the message locale in the `Key` field; for example, `en-gb`. 3. Enter the message to display to the user in the `Value` field. Default: none
One Time Password Length	The length of the generated OTP in digits. This value must be at least `6`. It must also be compatible with the hardware/software OTP generators you expect end users to use. For example, Google and ForgeRock authenticators support values of `6` and `8` respectively. Default: `6`

Property	Usage
Minimum Secret Key Length	Number of hexadecimal characters allowed for the secret key. Default: `32`
OATH Algorithm	Specify the algorithm the device uses to generate the OTP: ***HOTP*** HOTP uses a counter; the counter increments every time a new OTP is generated. When you use this setting, also set the same value in the OATH Token Verifier node. ***TOTP*** TOTP generates a new OTP every few seconds as specified by the **TOTP Time Step Interval** setting. Default: `TOTP`
TOTP Time Step Interval	The length of time that an OTP is valid in seconds. For example, if the time step interval is 30 seconds, a new OTP is generated every 30 seconds and is valid for 30 seconds only. Default: `30` seconds
TOTP Hash Algorithm	The HMAC hash algorithm used to generate the OTP codes. AM supports SHA1, SHA256, and SHA512. Default: `SHA1`
HOTP Checksum Digit	This adds a digit to the end of the OTP generated to be used as a checksum to verify the OTP was generated correctly. This is in addition to the actual password length. Only set this if the user devices support it. Default: false
HOTP Truncation Offset	This is an option used by the HOTP algorithm that not all devices support. Leave the default value unless you know user devices use an offset. Default: `-1`

Property	Usage

Store device data in shared state	If enabled, the device data isn't stored in the user profile on successful completion of the node. Instead, the node adds the device data as a base64-encoded string to the `oathDeviceProfile` property in the shared node state. This string is decoded as an unescaped plain string representation of a JSON object. For example:
	In the shared node state:
	`oathDeviceProfile="eyAidXVpZCI6ICJhNDhiMjUyMS0xYzliLTRiYTct...ja0RyaWZ0U2Vjb25kcyI6IDAgfQ"`
	Decoded value:
	`{`
	`  "uuid": "a48b2521-1c9b-4ba7-a45c-8dd855c7397c",`
	`  "recoveryCodes": [],`
	`  "sharedSecret": "0CF9910A24CAF84E81CEBA71C2086DE4",`
	`  "deviceName": "OATH Device",`
	`  "lastLogin": 0,`
	`  "counter": 0,`
	`  "checksumDigit": false,`
	`  "truncationOffset": -1,`
	`  "clockDriftSeconds": 0`
	`}`
	Use the OATH Device Storage node to store the device data in the user profile instead.
	Default: false

Outputs

If the **Store device data in shared state** setting is enabled, this node records the device profile in the `oathDeviceProfile` shared state attribute.

If the **Generate Recovery Codes** setting is enabled, this node records the recovery codes in the `oathEnableRecoveryCode` shared state attribute.

Outcomes

`Success`

Device registration succeeded.

`Failure`

Any other case.

Errors

This node logs the following error messages:

`No username found.`

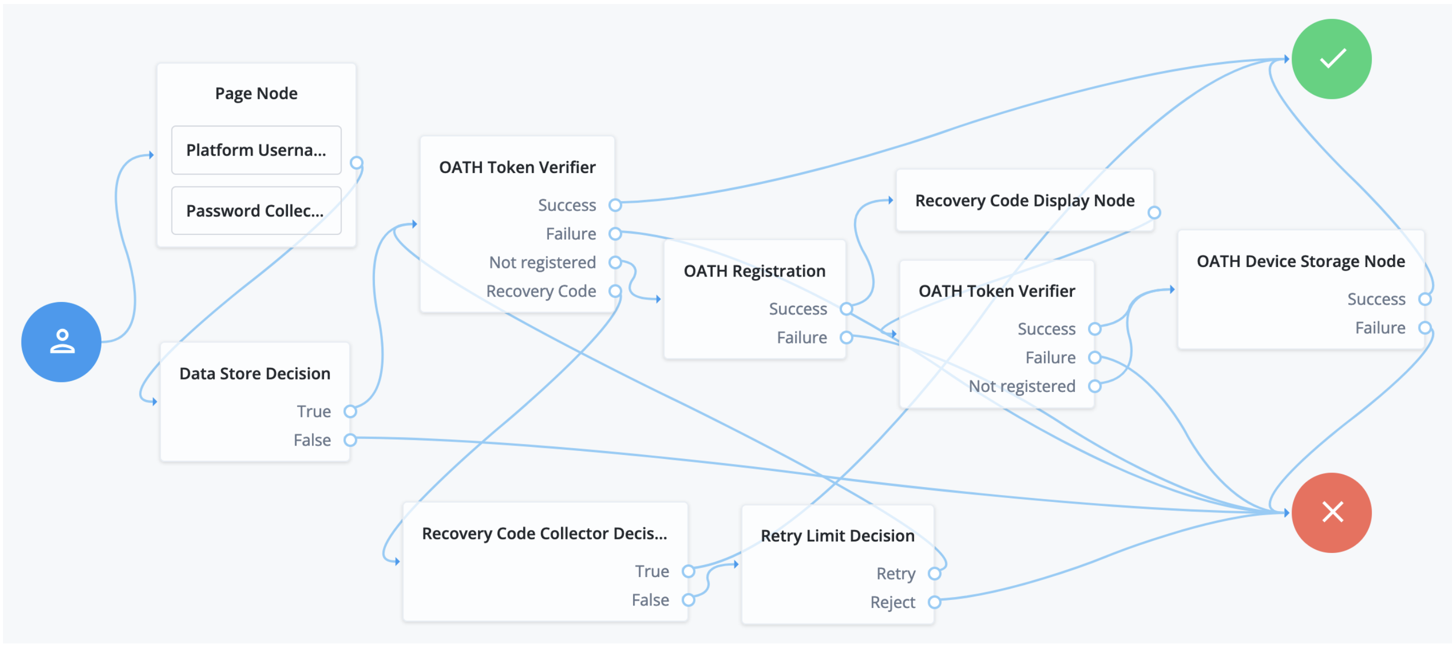
The node failed to read the username from the shared state.

`No device profile found on shared state`

The node failed to read the device profile from the shared state.

Example

The following journey includes both username-password and one-time passcode authentication:



- The Page node with the Platform Username node and the Platform Password node prompts for the user credentials.
- The Data Store Decision node confirms the username-password credentials.
- The first OATH Token Verifier node prompts for a one-time passcode with an option to use a recovery code.
- The OATH Registration node prompts the user to register a device and includes its profile in the shared state.
- The Recovery Code Display node shows the recovery codes and prompts the user to keep them safe.
- The second OATH Token Verifier node prompts for a one-time passcode using the newly registered device.
- The OATH Device Storage node writes the device profile to the user's account.
- The Recovery Code Collector Decision node prompts for a recovery code.
- The Retry Limit Decision node lets the user retry another code if they enter one incorrectly.

OATH Token Verifier node

The **OATH Token Verifier** node requests and verifies a one-time password (OTP) generated by a device such as a mobile phone.

The default configuration is time-based OTP (TOTP), but the node also supports HMAC (HOTP).

The node requires prior authentication and a device registered with an OATH Registration node.

Note

You can use the OATH nodes in conjunction with the ForgeRock Authenticator application to register your device, receive notifications, and generate one-time passwords.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Authenticators

The OATH-related nodes can integrate with the following authenticator apps:

- The ForgeRock Authenticator app for Android and iOS.
- Third-party authenticator apps that support the following open standards:
  - RFC 4226: HMAC-Based One-Time Password (HOTP)
  - RFC 6238: Time-Based One-Time Password (TOTP)

Inputs

This node reads the `username` attribute from the shared state.

Dependencies

Confirm the user credentials before letting them authenticate with a device. For example, precede this node with the following nodes earlier in the authentication flow:

- Username Collector node (standalone AM) or Platform Username node (Ping Identity Platform deployment)
- Password Collector node (standalone AM) or Platform Password node (Ping Identity Platform deployment)
- Data Store Decision node

Configuration

Property	Usage
OATH Algorithm	Specify the algorithm the device uses to generate the OTP: ***HOTP*** HOTP uses a counter; the counter increments every time a new OTP is generated. When you use this setting, also set the same value in the OATH Registration node. ***TOTP*** TOTP generates a new OTP every few seconds as specified by the **TOTP Time Step Interval** setting. Default: `TOTP`
HOTP Window Size	Specify how much the OTP device and the server counter can be out of sync. For example, if the window size is 100 and the server's last successful login was at counter value 2, the server accepts an OTP that is generated between counter 3 and 102. Default: `100`
TOTP Time Step Interval	The length of time an OTP is valid in seconds. For example, if the time step interval is 30 seconds, a new OTP is generated every 30 seconds and is valid for 30 seconds only. Default: `30` seconds
TOTP Time Steps	Specify how many time steps the OTP can be out of sync. This applies to codes generated before or after the current code. For example, with a time step of 1, the server accepts the previous, current, and next codes. Default: `2`
TOTP Hash Algorithm	The HMAC hash algorithm used to generate the OTP codes. The ForgeRock Authenticator application supports SHA1, SHA256, and SHA512. Default: `SHA1`
TOTP Maximum Allowed Clock Drift	Specify how many time steps the authenticator application can be out of sync with the server before manual resynchronization is required. For example, with **TOTP Time Steps** of `3` and a **TOTP Time Step Interval** of 30 (seconds), the server treats codes up to 90 seconds from the current time as belonging to the current time step. The drift for a user's device is calculated each time they enter a new code. If the drift exceeds this value, the outcome is `Failure`. Default: `5`
Allow recovery codes	If enabled, lets users provide a recovery code to authenticate. Default: false

Outputs

If the outcome is `Not registered`, this node sets `"mfaMethod": "oath"` in the shared state.

Outcomes

`Success`

The user has a registered device and the token code was verified.

`Failure`

The user was not authenticated, or the collected token code can't be verified.

`Not registered`

The user account has no registered device profiles.

`Recovery Code`

Allow recovery codes is enabled, and the user chose to provide a recovery code.

Errors

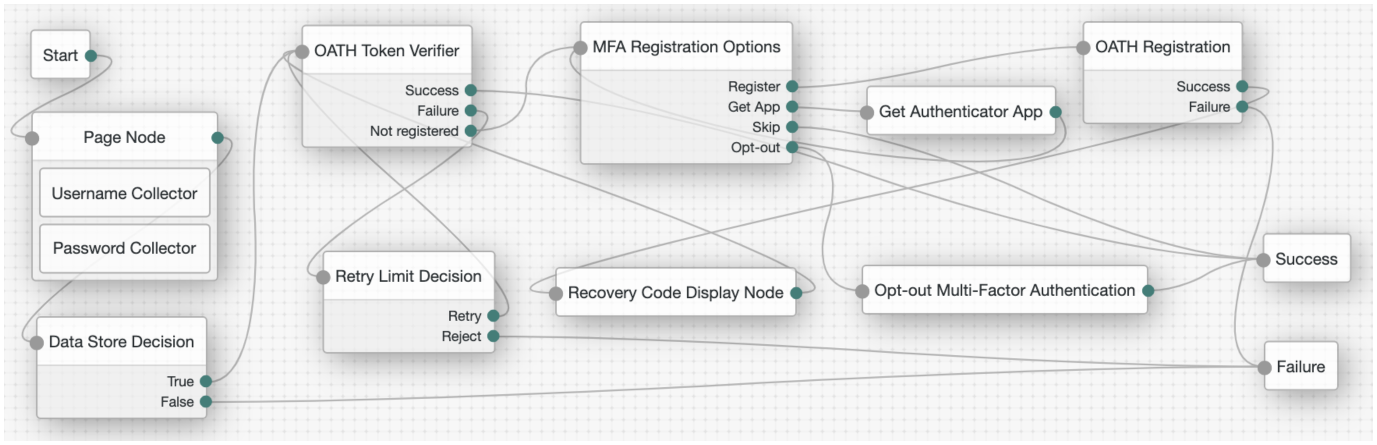
If this node cannot read the username from the shared state, it logs an error message: `Expected username to be set.`

If processing raises an exception, this node logs the detail as an error message.

Example

The following journeys use this node as part of a flexible multi-factor authentication (MFA) authentication flow:

PingAM standalone

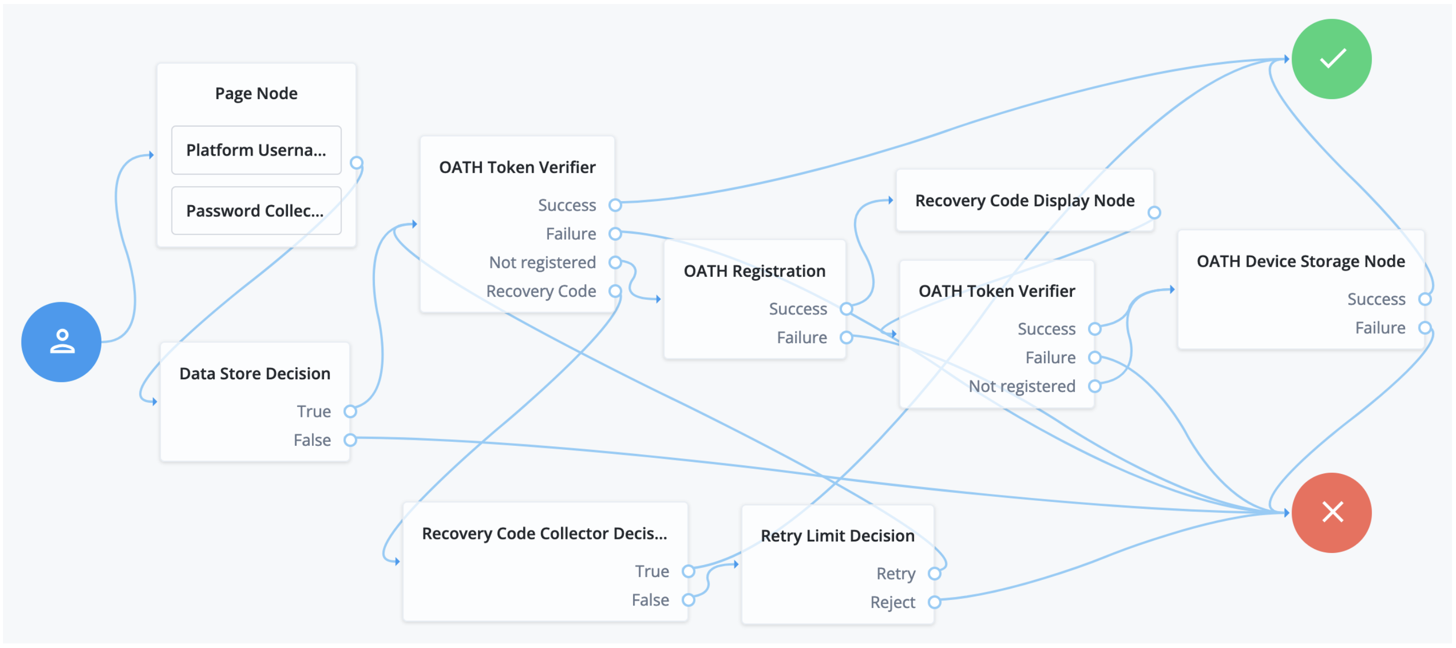


- The Page node with the Username Collector node and the Password Collector node prompts for the user credentials.
- The Data Store Decision node confirms the username-password credentials.

- The OATH Token Verifier node prompts for a one-time passcode with an option to use a recovery code.
- The Retry Limit Decision node lets the user retry another code if they enter one incorrectly.
- The MFA Registration Options node lets the user choose how to register their device.
- The Get Authenticator App node helps the user install the ForgeRock Authenticator application.
- The OATH Registration node prompts the user to register a device and includes its profile in the shared state.
- The Recovery Code Display node shows the recovery codes and prompts the user to keep them safe.
- The Opt-out Multi-Factor Authentication node lets the user choose to skip the second authentication factor.

Ping Identity Platform deployment

The following journey includes both username-password and one-time passcode authentication:



- The Page node with the Platform Username node and the Platform Password node prompts for the user credentials.
- The Data Store Decision node confirms the username-password credentials.
- The first OATH Token Verifier node prompts for a one-time passcode with an option to use a recovery code.
- The OATH Registration node prompts the user to register a device and includes its profile in the shared state.
- The Recovery Code Display node shows the recovery codes and prompts the user to keep them safe.
- The second OATH Token Verifier node prompts for a one-time passcode using the newly registered device.
- The OATH Device Storage node writes the device profile to the user's account.
- The Recovery Code Collector Decision node prompts for a recovery code.
- The Retry Limit Decision node lets the user retry another code if they enter one incorrectly.

Opt-out Multi-Factor Authentication node

The **Opt-out Multi-Factor Authentication** node sets an attribute in the user's profile, which records their decision to skip multi-factor authentication (MFA) on the selected device.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

- This node requires the `realm` and `username` properties in the incoming node state.

Implement a Username Collector node (standalone AM) or Platform Username node (Advanced Identity Cloud and Ping Identity Platform deployments) earlier in the journey.

- This node requires the `mfaMethod` in the incoming state to know what type of MFA device to update:
    - For push authentication, this node requires the `pushMessageId` in the incoming state, which is a unique ID to identify the push notification request.

      Implement a Push Sender node earlier in the journey.
    - For OATH authentication, implement the OATH Token Verifier node earlier in the journey.

Dependencies

This node has no dependencies.

Configuration

This node has no configurable properties.

Outputs

This node doesn't change the shared state.

This node updates the user's profile with either the `push2faEnabled` or `oath2faEnabled` attribute to record their decision to opt out.

These are the default attributes but they can be changed in the ForgeRock Authenticator (Push) service or the ForgeRock Authenticator (OATH) service.

Outcomes

Single outcome path.

Errors

The node can log the following errors:

- `Expected username to be set`

  The node can't identify the user from the shared state.

- `Expected MFA method to be set`

  The node can't identify the MFA method from the shared state.

- `Unable to set user attribute as skippable`

  The node can't set the relevant attribute for the user's current device.

- `Failed to get the identity object`

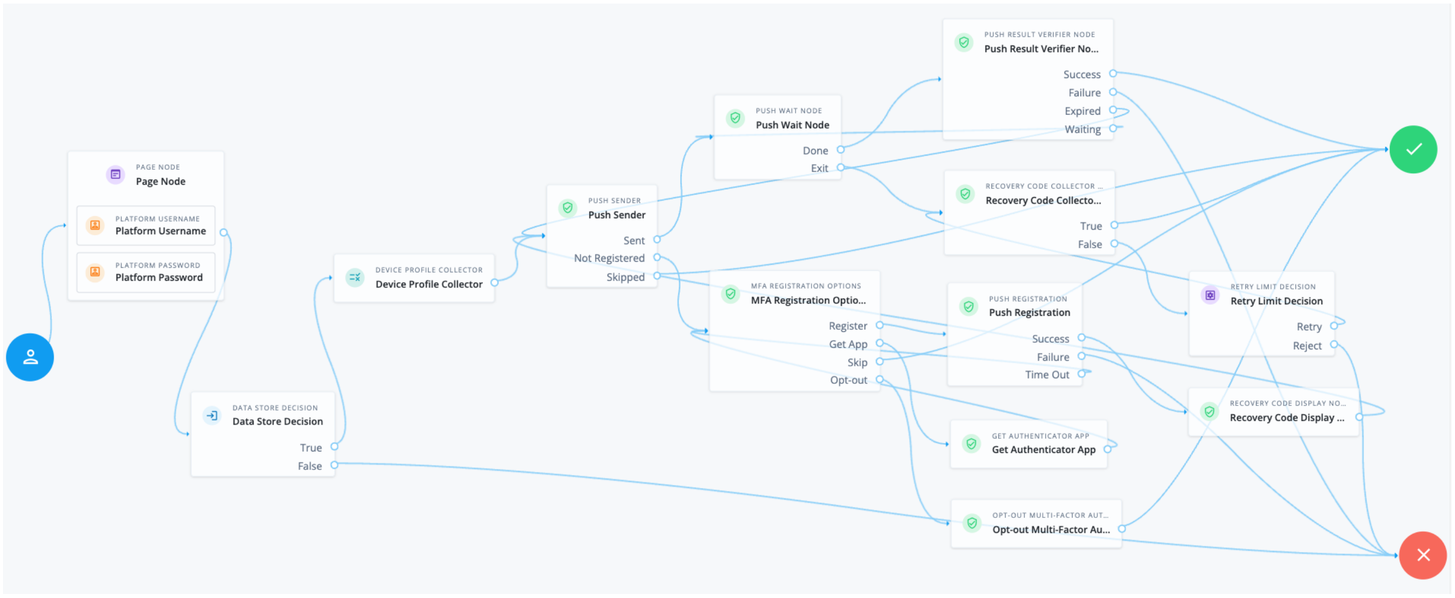
  The node can't read the identity of the account.

- `Unsupported MFA method has been set`

  The node can't retrieve the device profile details or the user has an unsupported MFA method set.

Examples

The following example shows one possible implementation of multi-factor push authentication, which uses this node:



List of node connections

Source node	Outcome path	Target node
Page Node containing nodes to collect credentials. For standalone AM deployments, implement a Username Collector node and a Password Collector node. For Advanced Identity Cloud and Ping Identity Platform deployments, implement a Platform Username node and a Platform Password node.	→	Data Store Decision
Data Store Decision	True	Device Profile Collector
	False	Failure
Device Profile Collector	→	Push Sender
Push Sender	Sent	Push Wait
	Not Registered	MFA Registration Options
	Skipped	Success
Push Wait	Done	Push Result Verifier
	Exit	Recovery Code Collector Decision
Push Result Verifier	Success	Success
	Failure	Failure
	Expired	Push Sender
	Waiting	Push Wait
MFA Registration Options	Register	Push Registration
	Get App	Get Authenticator App
	Skip	Success
	Opt-out	Opt-out Multi-Factor Authentication
Recovery Code Collector Decision	True	Success
	False	Retry Limit Decision

Source node	Outcome path	Target node
Push Registration	Success	Recovery Code Display Node
	Failure	Failure
	Time Out	MFA Registration Options
Get Authenticator App	→	MFA Registration Options
Opt-out Multi-Factor Authentication	→	Success
Retry Limit Decision	Retry	Recovery Code Collector Decision
	Reject	Failure
Recovery Code Display Node	→	Push Sender

After verifying the user's credentials, evaluation continues to the Device Profile Collector node to collect the device's location and then proceeds to the Push Sender node.

If the user *has* a registered device:

1. The Push Sender node sends a push notification to their registered device.
2. The Push Wait node pauses authentication for five seconds. During this time, the user can respond to the push notification on their device using the ForgeRock Authenticator app.

   If the user exits the Push Wait node, they're directed to the Recovery Code Collector Decision node, where they can enter a recovery code to authenticate.

   Tip

   Configure the **Exit Message** property in the Push Wait node with a message, such as `Lost phone? Use a recovery code` for situations like this.

   A Retry Limit Decision node allows three attempts to enter a recovery code before failing the authentication.
3. The Push Result Verifier node verifies the user's response:
   - If the user responds positively, they're authenticated successfully and logged in.
   - If the user responds negatively, authentication fails.
   - If the push notification expires, the Push Sender node sends a new push notification.

     Tip

     Use a Retry Limit Decision node to constrain the number of times a new code is sent.
   - If the user hasn't yet responded, the flow loops back a step and the Push Wait node pauses authentication for another 5 seconds.

If the user *doesn’t have* a registered device:

1. The MFA Registration Options node presents the user with the following options:

   Register Device

   The flow continues to the Push Registration node, which displays a QR code for the user to scan with their authenticator app.

   Get the App

   Displayed only if the node is configured to display Get Authenticator App. The flow continues to the Get Authenticator App node, which displays links to download the authenticator app.

   Skip this step

   Displayed only if the node is configured to allow users to skip registration. In this example, skipping is linked to the `Success` outcome. However, you could provide an alternative authentication flow using an Inner Tree Evaluator node for example.

   Opt-out

   Displayed only if the node is configured to allow users to skip registration. Evaluation continues to the Opt-out Multi-Factor Authentication node, which updates the user’s profile to skip MFA with push in the future. In this example, after updating the profile, the flow continues to the `Success` outcome.

2. The user registers the device with the Push Registration node.

   After registration, the Recovery Code Display node displays the recovery codes to the user and the flow returns to the Push Sender node to continue push authentication.

Note

To manage push devices, the user must log in using either the device or a recovery code.
Find more information in the MFA documentation for:

- PingAM
- Advanced Identity Cloud

OTP Collector Decision node

Requests and verifies one-time passwords.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes

Product	Available?
Ping Identity Platform (self-managed)	Yes

Outcomes

- `True`
- `False`

Evaluation continues along the `True` outcome path if the one-time password is valid; otherwise, evaluation continues along the `False` outcome path.

Properties

Property	Usage
One Time Password Validity Length	Specify the length of time, in minutes, that a one-time password remains valid. Default: `5`

OTP Email Sender node

The OTP Email Sender node sends an email containing a generated one-time password (OTP) to the user.

Send mail requests time out after 10 seconds.

Tip

You can change the timeout in the following advanced AM server properties:

- `org.forgerock.openam.smtp.system.connect.timeout`
- `org.forgerock.openam.smtp.system.socket.read.timeout`
- `org.forgerock.openam.smtp.system.socket.write.timeout`

- To configure advanced server properties for all the instances of the AM environment, go to **Configure** > **Server Defaults** > **Advanced** in the AM admin UI.
- To configure advanced server properties for a specific instance, go to **Deployment** > **Servers** > ***Server Name*** > **Advanced**.

If the property you want to add or edit is already configured, click the pencil (✎) button to edit it, then click the checkmark (✓) button.
Save your changes.
For more information, refer to advanced properties.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Prerequisites

- The node requires a configured email provider.

Inputs

This node requires the following input from the shared state:

- The authenticating user's ID. The node queries the user's entry for an email address.

  Implement an Attribute Collector node node before this node to obtain the user's ID.

- The OTP stored in the `oneTimePassword` transient state property.

  Implement the HOTP Generator node before this node in the journey to obtain the OTP.

Configuration

Property	Usage
Mail Server Host Name *(required)*	The hostname of the SMTP email server.
Mail Server Host Port	The outgoing mail server port. Common ports are 25, 465 for SSL/TLS, or 587 for StartTLS.
Mail Server Authentication Username	The username AM uses to connect to the mail server.
Mail Server Authentication Password	The password AM uses to connect to the mail server. **Note** This property is deprecated. Use the **Mail Server Secret Label Identifier** instead. If you set a **Mail Server Secret Label Identifier**, this password is ignored.

Property	Usage
Mail Server Secret Label Identifier	An identifier used to create a *secret label* for mapping to a secret in a secret store. AM uses this identifier to create a specific secret label for this node. The secret label takes the form `am.authentication.nodes.otp.mail.identifier.password` where identifier is the value of **Mail Server Secret Label Identifier**. The identifier can only contain alphanumeric characters `a-z`, `A-Z`, `0-9`, and periods (`.`). It can't start or end with a period. If you set a **Mail Server Secret Label Identifier** and AM finds a matching secret in a secret store, the **Mail Server Authentication Password** is ignored.
Email From Address *(required)*	The email address from which the OTP will appear to have been sent.
Email Attribute Name	The attribute in the user profile that contains the email address to which the email with the OTP is sent. Default: `mail`
The subject of the email	Click **Add** to add a new email subject. Enter the locale, such as `en-uk`, in the **Key** field and the subject in the **Value** field. Repeat these steps for each locale that you support.
The content of the email	Click **Add** to add the content of the email. Enter the locale, such as `en-uk`, in the **Key** field and the email content in the **Value** field. Repeat these steps for each locale that you support.
Mail Server Secure Connection	Set the connection method to the mail server. If you set a secure method here, AM must trust the server certificate of the mail server. The possible values for this property are: • `NON SSL/TLS` • `SSL/TLS` • `Start TLS` Default: `SSL/TLS`
Gateway Implementation Class	The class the node uses to send SMS and email messages. A custom class must implement the `com.sun.identity.authentication.modules.hotp.SMSGateway` interface. Default: `com.sun.identity.authentication.modules.hotp.DefaultSMSGatewayImpl`

Outputs

This node copies shared and transient state into the outgoing node state.

Errors

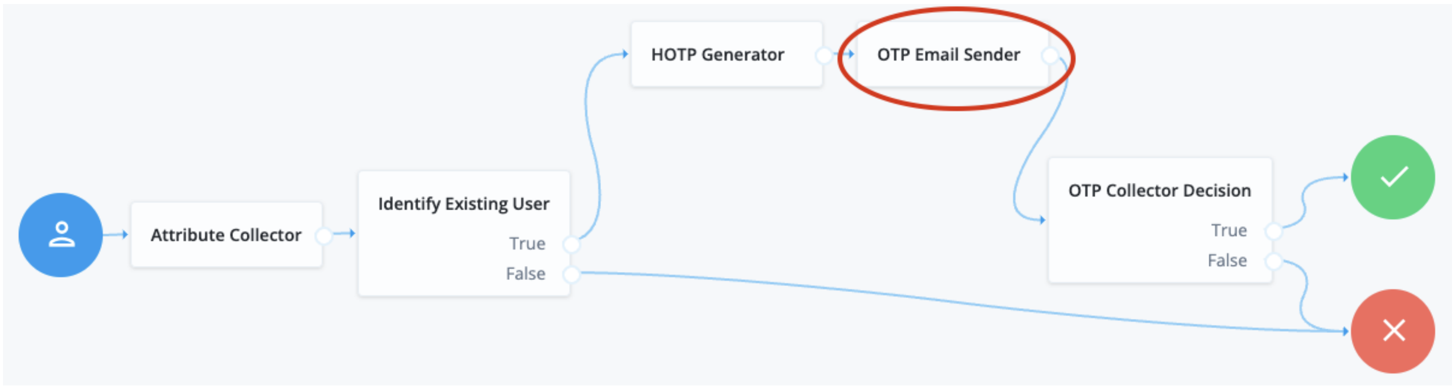
The node throws an `IdRepoException` and an `SSOException` error if it's unable to obtain the user's email address.

Outcomes

Single outcome path.

Implement an OTP Collector Decision node after this node to continue the authentication journey.

Example



OTP SMS Sender node

The OTP SMS Sender node uses an email-to-SMS gateway provider to send an SMS message containing a generated one-time password (OTP) to the user.

The node sends an email to an address formed by joining the following values together:

- The user's telephone number, obtained by querying a specified profile attribute, for example, `telephoneNumber`.
- The `@` character.
- The email-to-SMS gateway domain, obtained by querying the profile attribute specified by the **Mobile Carrier Attribute Name** property.

For example, if configured to use the *TextMagic* email-to-SMS service, the node might send an email through the specified SMTP server to the address: `18005550187@textmagic.com`.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Prerequisites

- The node requires a configured email-to-SMS gateway provider.

Inputs

This node requires the following input from the shared state:

- The authenticating user's ID. The node queries the user's entry for a telephone number.

  Implement an Attribute Collector node node before this node to obtain the user's ID.

- The OTP stored in the `oneTimePassword` transient state property.

  Implement the HOTP Generator node before this node in the journey to obtain the OTP.

Configuration

Property	Usage
Mail Server Host Name *(required)*	The hostname of the SMTP email server.
Mail Server Host Port	The outgoing mail server port. Common ports are 25, 465 for SSL/TLS, or 587 for StartTLS.
Mail Server Authentication Username	The username AM uses to connect to the mail server.
Mail Server Authentication Password	The password AM uses to connect to the mail server. **Note** This property is deprecated. Use the **Mail Server Secret Label Identifier** instead. If you set a **Mail Server Secret Label Identifier**, this password is ignored.
Mail Server Secret Label Identifier	An identifier used to create a *secret label* for mapping to a secret in a secret store. AM uses this identifier to create a specific secret label for this node. The secret label takes the form `am.authentication.nodes.otp.sms.identifier.password` where identifier is the value of **Mail Server Secret Label Identifier**. The identifier can only contain alphanumeric characters `a-z`, `A-Z`, `0-9`, and periods (`.`). It can't start or end with a period. If you set a **Mail Server Secret Label Identifier** and AM finds a matching secret in a secret store, the **Mail Server Authentication Password** is ignored.
Email From Address *(required)*	The email address from which the OTP will appear to have been sent.
Mobile Phone Number Attribute Name	The attribute in the user profile that contains the mobile phone number to which the SMS with the OTP is sent. Default: `telephoneNumber`

Property	Usage
Mobile Carrier Attribute Name	The attribute in the user profile that contains the mobile carrier domain for sending SMS messages. By default, an AM user profile doesn't have an attribute for the mobile carrier domain. You can customize the user profile by adding a new attribute to it, then populate that attribute with users' SMS messaging domains. All mobile carriers and bulk SMS messaging services have associated SMS messaging domains. For example, Verizon uses `vtext.com`, T-Mobile uses `tmomail.net`, and the TextMagic service uses `textmagic.com`. If you plan to send text messages internationally, determine whether the messaging service requires a country code. If you leave the **Mobile Carrier Attribute Name** property empty, AM defaults to sending SMS messages using `txt.att.net` for all users.
The subject of the message	Click **Add** to add a new message subject. Enter the locale, such as `en-uk`, in the **Key** field and the subject in the **Value** field. Repeat these steps for each locale that you support.
The content of the message	Click **Add** to add the content of the message. Enter the locale, such as `en-uk`, in the **Key** field and the email content in the **Value** field. Repeat these steps for each locale that you support.
Mail Server Secure Connection	Set the connection method to the mail server. If you set a secure method here, AM must trust the server certificate of the mail server. The possible values for this property are: • `NON SSL/TLS` • `SSL/TLS` • `Start TLS` Default: `SSL/TLS`
Gateway Implementation Class	The class the node uses to send SMS and email messages. A custom class must implement the `com.sun.identity.authentication.modules.hotp.SMSGateway` interface. Default: `com.sun.identity.authentication.modules.hotp.DefaultSMSGatewayImpl`

Outputs

This node copies shared and transient state into the outgoing node state.

Errors

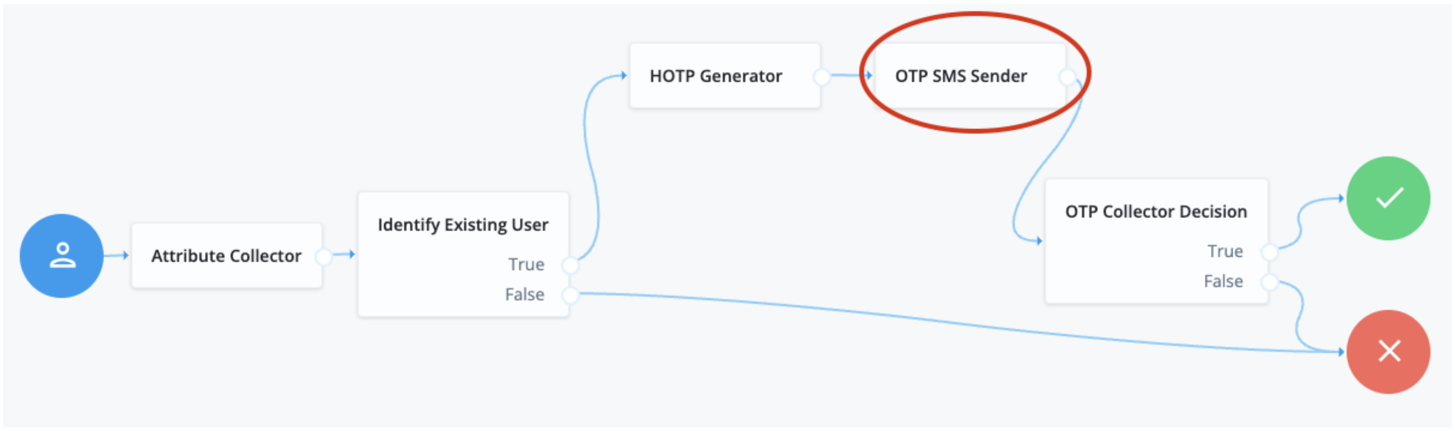
The node throws an `IdRepoException` and an `SSOException` error if it's unable to obtain the user's telephone number.

Outcomes

Single outcome path.

Implement an OTP Collector Decision node after this node to continue the authentication journey.

Example



Push Registration node

The **Push registration** node lets a user register their device, such as a mobile phone for multi-factor authentication (MFA) using push notifications.

Learn more in the Push Authentication documentation for:

- Advanced Identity Cloud
- PingAM

Tip

You can use the Combined MFA Registration node to register a device for use with both push notifications and one-time passcode (OATH) verification in a single step.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Authenticators

The push-related nodes integrate with the ForgeRock Authenticator app for Android and iOS.

Third-party authenticator apps aren't compatible with the push notification functionality.

Inputs

This node requires the `realm` and `username` properties in the incoming node state.

Implement a Username Collector node (standalone AM) or Platform Username node (Advanced Identity Cloud and Ping Identity Platform deployments) earlier in the journey.

Dependencies

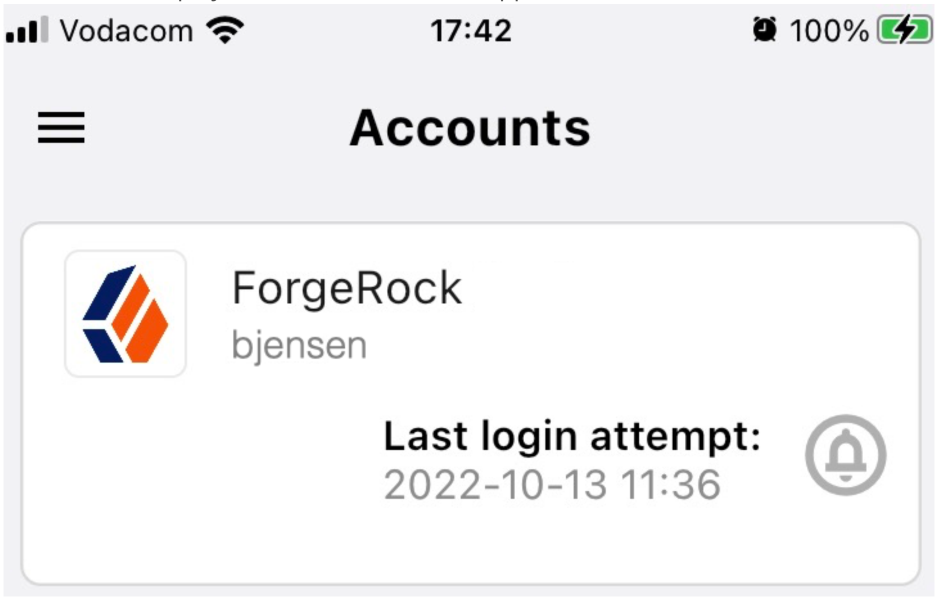
You must configure the Push Notification service for the realm to use this node. Optionally, also configure the ForgeRock Authenticator (Push) service.

Find more information in the corresponding documentation for:

- Advanced Identity Cloud
- PingAM

Find information on provisioning the credentials used by the service in How To Configure Service Credentials (Push Auth, Docker) in Backstage.

Configuration

Property	Usage
Issuer	The name of the issuer or application so the user knows which service their account relates to. This value is displayed in the authenticator app.  Vodacom 17:42 100% Accounts ForgeRock bjensen Last login attempt: 2022-10-13 11:36
Account Name	The profile attribute to display as the username in the authenticator app. If not specified, or if the specified profile attribute is empty, their username is used.
Background Color	The background color, in hex notation, to display behind the issuer's logo within the authenticator app.
Logo Image URL	The location of an image to download and display as the issuer's logo in the authenticator app.
Generate Recovery Codes	Select this option to generate push-specific recovery codes. When enabled, recovery codes are generated and stored in the transient state if registration is successful. Use the Recovery Code Display node to display the codes to the user for safe keeping. **Important** Generating recovery codes overwrites all existing push-specific recovery codes. Only the most recent set of recovery codes can be used for authentication if a device is lost or stolen.

Property	Usage
QR code message	(Optional) Add a custom, localized message to display to the user with instructions to scan the QR code to register the device: 1. Click **+**. 2. In the **Key** field, enter the locale. For example, `en-gb`.[(1)] 3. In the **Value** field, enter the message. 4. Click **Done**. 5. Repeat to add more messages and save your changes when you're done. Leave blank to use the default message.[(2)] Default: `Open your Authenticator app and tap the number shown to sign-in`
Registration Response Timeout	The number of seconds to wait for a response from the authenticator app. As soon as the specified time is reached, evaluation continues along the `Time Out` outcome path.

[(1)] Specify a locale that Java supports, such as `en-gb`. Otherwise, the node throws a configuration exception with an `Invalid locale provided` message.

[(2)] PingAM only: Learn more about customizing and translating default messages in Internationalize nodes.

Outputs

- The node adds the `pushDeviceProfiles` attribute to the user's profile with the device details on successful registration.
- The node updates the shared state with the push device settings, the message ID and the push challenge.
- If **Generate Recovery Codes** is enabled, the node records the recovery codes in the `recoveryCodes` shared state attribute. Use the Recovery Code Display node to display the codes to the user for safe keeping.

Outcomes

`Success`

The user successfully registered their authenticator app.

`Failure`

An issue occurred during device registration.

`Time Out`

A response wasn't received from the user's device within the time specified in the node configuration.

Errors

The node can log the following errors:

- `Unable to find push message ID`

The node failed to read the `pushMessageId` from the shared state and can't proceed with registration. Make sure you have implemented a Push Sender node earlier in the journey.

- `Expected username to be set`

  The node can't identify the user from the shared state.

- `Unable to read service addresses for Push Notification Service`

  The node can't retrieve the push service URLs. Check that the Push Notification service is set up correctly in the realm.

- `Could not get messageId`

  The node fails to retrieve the messageId and can't proceed with registration.

- `The push message corresponds to <message type> message type which is not registered in the <realm name> realm`

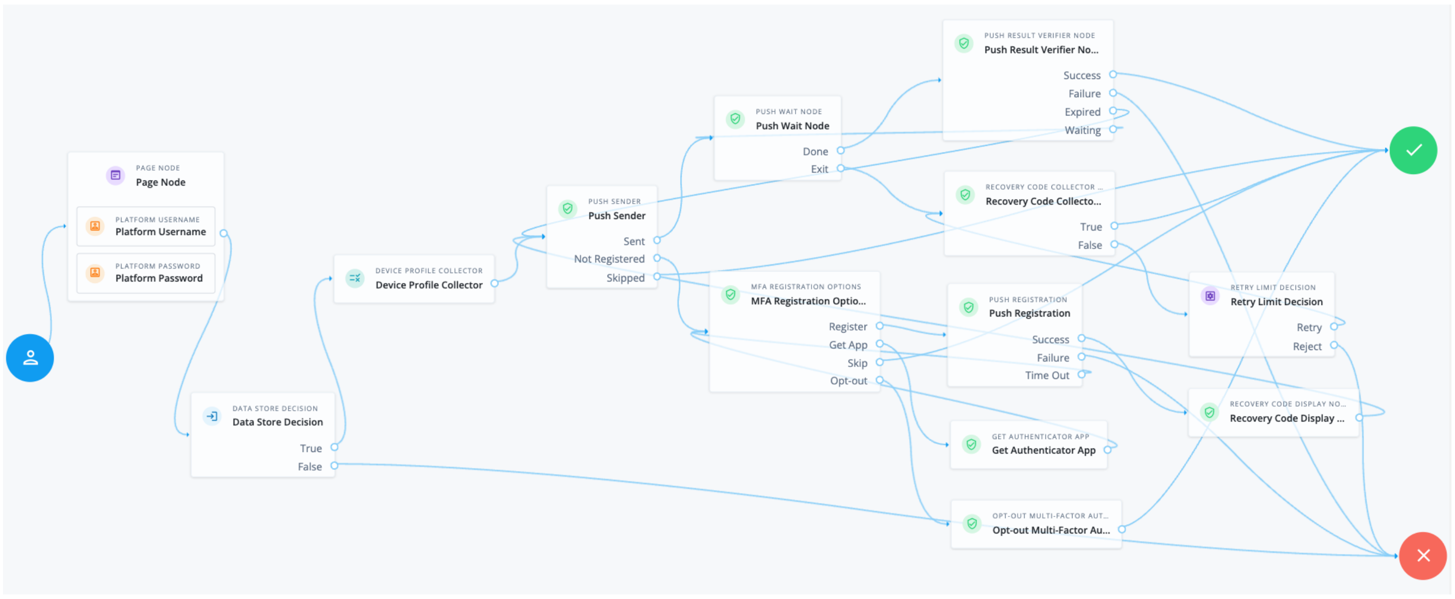
  The node can't start the registration process. Check that the Push Notification service is set up correctly in the realm.

- `Failed to save device to user profile`

  The node can't save the device details to the user's profile.

Examples

The following example shows one possible implementation of multi-factor push authentication, which uses this node:



List of node connections

Source node	Outcome path	Target node
Page Node containing nodes to collect credentials. For standalone AM deployments, implement a Username Collector node and a Password Collector node. For Advanced Identity Cloud and Ping Identity Platform deployments, implement a Platform Username node and a Platform Password node.	→	Data Store Decision
Data Store Decision	True	Device Profile Collector
	False	Failure
Device Profile Collector	→	Push Sender
Push Sender	Sent	Push Wait
	Not Registered	MFA Registration Options
	Skipped	Success
Push Wait	Done	Push Result Verifier
	Exit	Recovery Code Collector Decision
Push Result Verifier	Success	Success
	Failure	Failure
	Expired	Push Sender
	Waiting	Push Wait
MFA Registration Options	Register	Push Registration
	Get App	Get Authenticator App
	Skip	Success
	Opt-out	Opt-out Multi-Factor Authentication
Recovery Code Collector Decision	True	Success
	False	Retry Limit Decision

Source node	Outcome path	Target node
Push Registration	Success	Recovery Code Display Node
	Failure	Failure
	Time Out	MFA Registration Options
Get Authenticator App	→	MFA Registration Options
Opt-out Multi-Factor Authentication	→	Success
Retry Limit Decision	Retry	Recovery Code Collector Decision
	Reject	Failure
Recovery Code Display Node	→	Push Sender

After verifying the user's credentials, evaluation continues to the Device Profile Collector node to collect the device's location and then proceeds to the Push Sender node.

If the user *has* a registered device:

1. The Push Sender node sends a push notification to their registered device.
2. The Push Wait node pauses authentication for five seconds. During this time, the user can respond to the push notification on their device using the ForgeRock Authenticator app.

   If the user exits the Push Wait node, they're directed to the Recovery Code Collector Decision node, where they can enter a recovery code to authenticate.

   > **Tip**
   >
   > Configure the **Exit Message** property in the Push Wait node with a message, such as `Lost phone? Use a recovery code` for situations like this.

   A Retry Limit Decision node allows three attempts to enter a recovery code before failing the authentication.
3. The Push Result Verifier node verifies the user's response:
   - If the user responds positively, they're authenticated successfully and logged in.
   - If the user responds negatively, authentication fails.
   - If the push notification expires, the Push Sender node sends a new push notification.

     > **Tip**
     >
     > Use a Retry Limit Decision node to constrain the number of times a new code is sent.
   - If the user hasn't yet responded, the flow loops back a step and the Push Wait node pauses authentication for another 5 seconds.

If the user *doesn't have* a registered device:

1. The MFA Registration Options node presents the user with the following options:

   Register Device

   The flow continues to the Push Registration node, which displays a QR code for the user to scan with their authenticator app.

   Get the App

   Displayed only if the node is configured to display Get Authenticator App. The flow continues to the Get Authenticator App node, which displays links to download the authenticator app.

   Skip this step

   Displayed only if the node is configured to allow users to skip registration. In this example, skipping is linked to the `Success` outcome. However, you could provide an alternative authentication flow using an Inner Tree Evaluator node for example.

   Opt-out

   Displayed only if the node is configured to allow users to skip registration. Evaluation continues to the Opt-out Multi-Factor Authentication node, which updates the user's profile to skip MFA with push in the future. In this example, after updating the profile, the flow continues to the `Success` outcome.

2. The user registers the device with the Push Registration node.

   After registration, the Recovery Code Display node displays the recovery codes to the user and the flow returns to the Push Sender node to continue push authentication.

Note

To manage push devices, the user must log in using either the device or a recovery code.
Find more information in the MFA documentation for:

- PingAM
- Advanced Identity Cloud

Push Result Verifier node

The **Push Result Verifier** node validates the user's response to a previously sent push notification message.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes

Product	Available?
Ping Identity Platform (self-managed)	Yes

Authenticators

The push-related nodes integrate with the ForgeRock Authenticator app for Android and iOS.

Third-party authenticator apps aren't compatible with the push notification functionality.

Inputs

- This node requires the `realm` and `username` properties in the incoming node state.

Implement a Username Collector node (standalone AM) or Platform Username node (Advanced Identity Cloud and Ping Identity Platform deployments) earlier in the journey.

- This node also requires the `pushMessageId` in the incoming state, which is a unique ID to identify the push notification request.

  Implement a Push Sender node earlier in the journey.

Dependencies

You must configure the Push Notification service for the realm to use this node. Optionally, also configure the ForgeRock Authenticator (Push) service.

Find more information in the corresponding documentation for:

- Advanced Identity Cloud
- PingAM

Find information on provisioning the credentials used by the service in How To Configure Service Credentials (Push Auth, Docker) in Backstage.

Configuration

This node has no configurable properties.

Outputs

This node doesn't change the shared state.

Outcomes

Success

The user approved the push notification.

`Failure`

The user rejected the push notification.

`Expired`

A response wasn't received from the user's device within the time specified in the Push Sender node.

`Waiting`

A response hasn't been received yet.

Errors

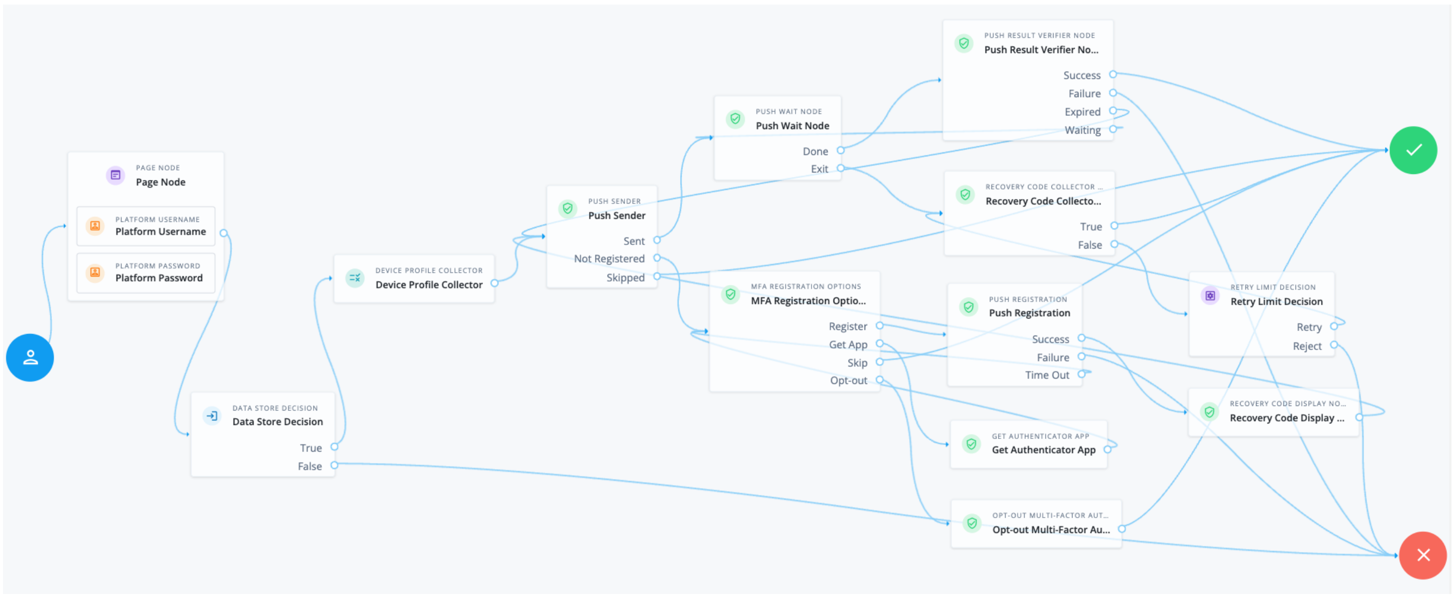
The node can log the following errors:

- `Unable to find push message ID`

  The node failed to read the `pushMessageId` from the shared state and can't verify the user's response. Make sure you have implemented a Push Sender node earlier in the journey.

Examples

The following example shows one possible implementation of multi-factor push authentication, which uses this node:



List of node connections

Source node	Outcome path	Target node
Page Node containing nodes to collect credentials. For standalone AM deployments, implement a Username Collector node and a Password Collector node. For Advanced Identity Cloud and Ping Identity Platform deployments, implement a Platform Username node and a Platform Password node.	→	Data Store Decision
Data Store Decision	True	Device Profile Collector
	False	Failure
Device Profile Collector	→	Push Sender
Push Sender	Sent	Push Wait
	Not Registered	MFA Registration Options
	Skipped	Success
Push Wait	Done	Push Result Verifier
	Exit	Recovery Code Collector Decision
Push Result Verifier	Success	Success
	Failure	Failure
	Expired	Push Sender
	Waiting	Push Wait
MFA Registration Options	Register	Push Registration
	Get App	Get Authenticator App
	Skip	Success
	Opt-out	Opt-out Multi-Factor Authentication
Recovery Code Collector Decision	True	Success
	False	Retry Limit Decision

Source node	Outcome path	Target node
Push Registration	Success	Recovery Code Display Node
	Failure	Failure
	Time Out	MFA Registration Options
Get Authenticator App	→	MFA Registration Options
Opt-out Multi-Factor Authentication	→	Success
Retry Limit Decision	Retry	Recovery Code Collector Decision
	Reject	Failure
Recovery Code Display Node	→	Push Sender

After verifying the user's credentials, evaluation continues to the Device Profile Collector node to collect the device's location and then proceeds to the Push Sender node.

If the user *has* a registered device:

1. The Push Sender node sends a push notification to their registered device.
2. The Push Wait node pauses authentication for five seconds. During this time, the user can respond to the push notification on their device using the ForgeRock Authenticator app.

   If the user exits the Push Wait node, they're directed to the Recovery Code Collector Decision node, where they can enter a recovery code to authenticate.

   > **Tip**
   >
   > Configure the **Exit Message** property in the Push Wait node with a message, such as `Lost phone? Use a recovery code` for situations like this.

   A Retry Limit Decision node allows three attempts to enter a recovery code before failing the authentication.
3. The Push Result Verifier node verifies the user's response:
   - If the user responds positively, they're authenticated successfully and logged in.
   - If the user responds negatively, authentication fails.
   - If the push notification expires, the Push Sender node sends a new push notification.

     > **Tip**
     >
     > Use a Retry Limit Decision node to constrain the number of times a new code is sent.
   - If the user hasn't yet responded, the flow loops back a step and the Push Wait node pauses authentication for another 5 seconds.

If the user *doesn't have* a registered device:

1. The MFA Registration Options node presents the user with the following options:

   Register Device

   The flow continues to the Push Registration node, which displays a QR code for the user to scan with their authenticator app.

   Get the App

   Displayed only if the node is configured to display Get Authenticator App. The flow continues to the Get Authenticator App node, which displays links to download the authenticator app.

   Skip this step

   Displayed only if the node is configured to allow users to skip registration. In this example, skipping is linked to the `Success` outcome. However, you could provide an alternative authentication flow using an Inner Tree Evaluator node for example.

   Opt-out

   Displayed only if the node is configured to allow users to skip registration. Evaluation continues to the Opt-out Multi-Factor Authentication node, which updates the user's profile to skip MFA with push in the future. In this example, after updating the profile, the flow continues to the `Success` outcome.

2. The user registers the device with the Push Registration node.

   After registration, the Recovery Code Display node displays the recovery codes to the user and the flow returns to the Push Sender node to continue push authentication.

Note

To manage push devices, the user must log in using either the device or a recovery code.
Find more information in the MFA documentation for:

- PingAM
- Advanced Identity Cloud

Push Sender node

The **Push Sender** node sends push notification messages to a device for multi-factor authentication (MFA).

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes

Product	Available?
Ping Identity Platform (self-managed)	Yes

Authenticators

The push-related nodes integrate with the ForgeRock Authenticator app for Android and iOS.

Third-party authenticator apps aren't compatible with the push notification functionality.

Inputs

- This node requires the `realm` and `username` properties in the incoming node state.

Implement a Username Collector node (standalone AM) or Platform Username node (Advanced Identity Cloud and Ping Identity Platform deployments) earlier in the journey.

- This node can read the device location from the incoming node state if it exists.

  The device location only exists when a Device Profile Collector node is implemented earlier in the journey with the **Collect Device Location** option selected.

Dependencies

You must configure the Push Notification service for the realm to use this node. Optionally, also configure the ForgeRock Authenticator (Push) service.

Find more information in the corresponding documentation for:

- Advanced Identity Cloud
- PingAM

Find information on provisioning the credentials used by the service in How To Configure Service Credentials (Push Auth, Docker) in Backstage.

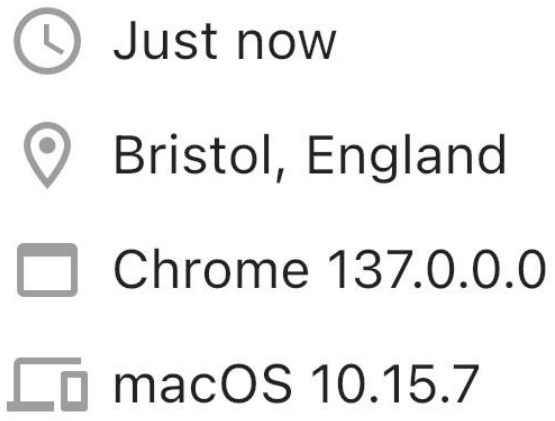
Configuration

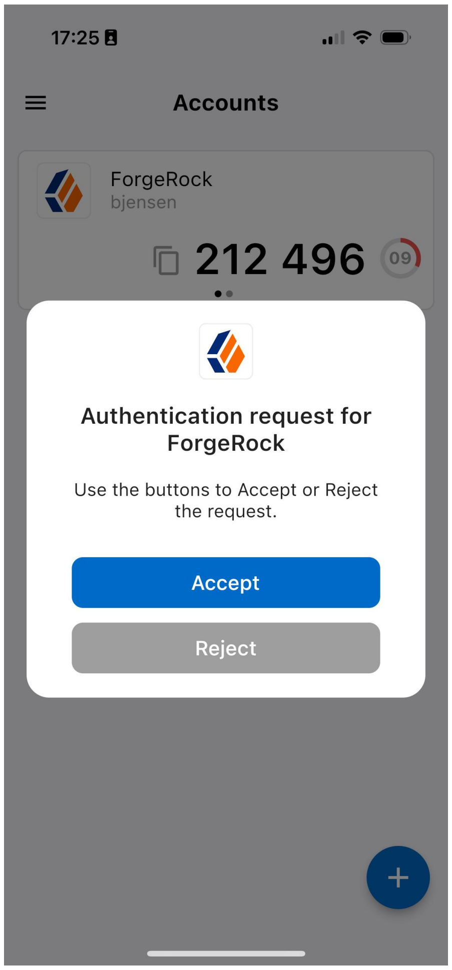
Property	Usage
Message Timeout	The number of milliseconds that the push notification message remains valid. The Push Result Verifier node rejects responses to push messages that have timed out. Default: `120000`

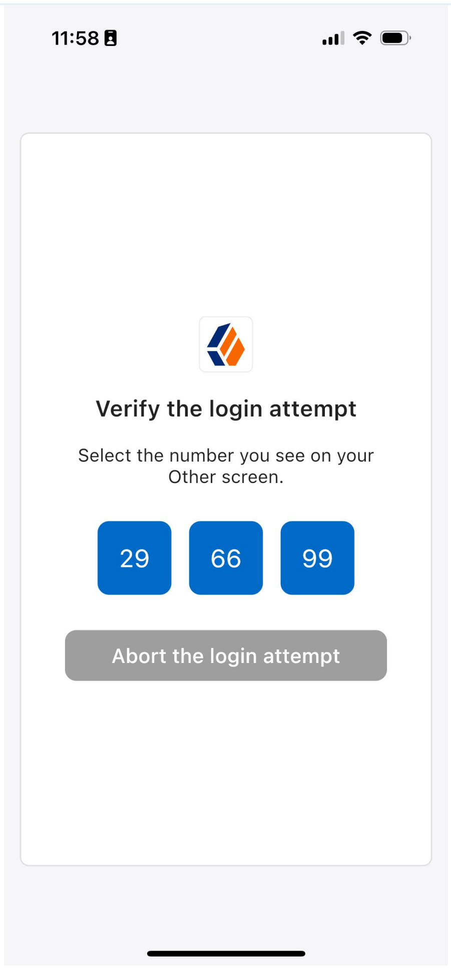
Property	Usage

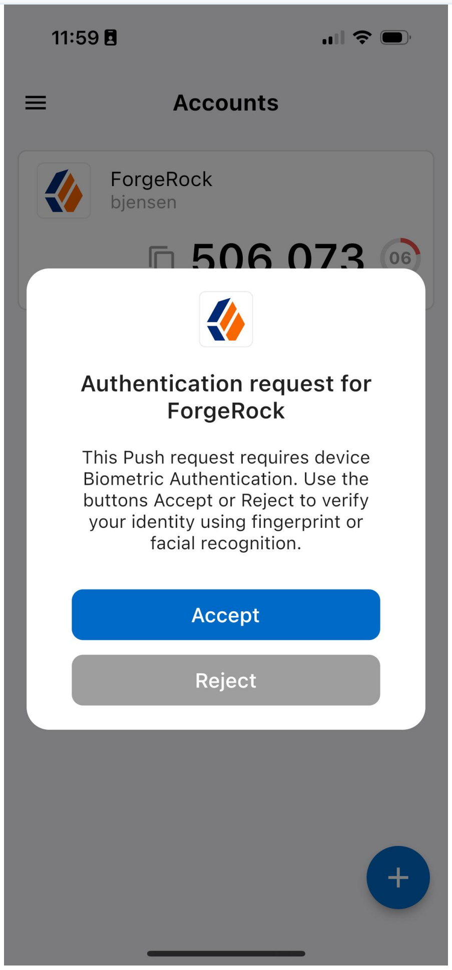
User Message	(Optional) Add a custom, localized message to send to the user. You can use the following variables in the **Value** field:
	
	`{{user}}`
	This variable is replaced with the username value obtained from the shared state, for example, `bjensen`.
	`{{issuer}}`
	This variable is replaced with the issuer value read from the device profile metadata, which is stored in the `pushDeviceProfiles` attribute by default.
	
	1. Click +.
	2. In the **Key** field, enter the locale. For example, `en-gb`.[1]
	3. In the **Value** field, enter the message.
	4. Click **Done**.
	5. Repeat to add more messages and save your changes when you're done.
	
	Leave blank to use the default message.[2]
	Default: `Login attempt from {{user}} at {{issuer}}`

Remove 'skip' option	Select this option to make push authentication mandatory. The `Skipped` outcome is not available when this option is selected. When disabled, the user can skip push authentication if required.

Property	Usage
Share Context Info	Select this option to include context data such as `remoteIp`, `userAgent`, and `location` in the notification payload. ```{ "location": { "latitude": 51.454514, "longitude": -2.587910 }, "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36", "remoteIp": "9.9.9.9" }``` The ForgeRock Authenticator app displays this additional information to the user to help them verify that the request is genuine and initiated by them. For example:  Just now Bristol, England Chrome 137.0.0.0 macOS 10.15.7 To include the `location` attribute, the journey must include a Device Profile Collector node with the **Collect Device Location** option selected.
Custom Payload Attributes	(Optional) Enter the names of the shared state objects to include in the message payload sent to the client. Enter each name separately and press Enter to add it. The size of the payload mustn't exceed 3 Kb.

Push Type	Select the type of push authentication the user must perform on their device to continue the journey. Possible values are: **`Tap to Accept`** ***(default)*** The user must tap `Accept` on their device to verify the request, or tap `Reject`.  Authentication request for ForgeRock Use the buttons to Accept or Reject the request. 17:25 Accounts ForgeRock bjensen 212 496 09 Accept Reject **Note** Research shows that users might accept a push authentication without checking it's legitimate. To reduce the chances of a user accepting a malicious push authentication attempt, consider using `Display Challenge Code` or `Use Biometrics to Accept` instead. **`Display Challenge Code`** The user must select one of three numbers displayed on their device. The number they select must match the code displayed in the browser to verify the request.

	 11:58 Verify the login attempt Select the number you see on your Other screen. 29 66 99 Abort the login attempt **`Use Biometrics to Accept`** The user must tap `Accept` on their device and then authenticate using biometrics to verify the request.

Property	Usage
	 11:59 Accounts ForgeRock bjensen 506 073 06 **Authentication request for ForgeRock** This Push request requires device Biometric Authentication. Use the buttons Accept or Reject to verify your identity using fingerprint or facial recognition.
Capture failure	Select this option to store the failure reason in the `PushAuthFailureReason` shared state attribute when the node fails to send the push notification. The `Failure` outcome is only available when this option is selected.

[1] Specify a locale that Java supports, such as `en-gb`. Otherwise, the node throws a configuration exception with an `Invalid locale provided` message.

[2] PingAM only: Learn more about customizing and translating default messages in Internationalize nodes.

Outputs

- The node adds a unique ID to identify the push notification request to the `pushMessageId` shared state attribute.
- If the outcome is `Not Registered`, this node sets `"mfaMethod": "push"` in the shared state.
- If the node fails to send the push notification and **Capture failure** is enabled, the node adds the failure reason to a property named `PushAuthFailureReason` in the shared state. Other nodes can read this property later in the journey, if required.

Possible failure reasons are:

- `MISSING_USERNAME`
- `SENDER_ALREADY_USED`
- `CTS_ERROR`
- `TRANSMISSION_FAILURE`

Outcomes

`Sent`

The push notification was sent successfully to the device.

`Not Registered`

The user doesn't have a registered device.

`Skipped`

The user chooses to skip push authentication.

`Failure`

An error occurred during node execution.

Errors

The node can log the following errors:

- `Failed to fetch identity`

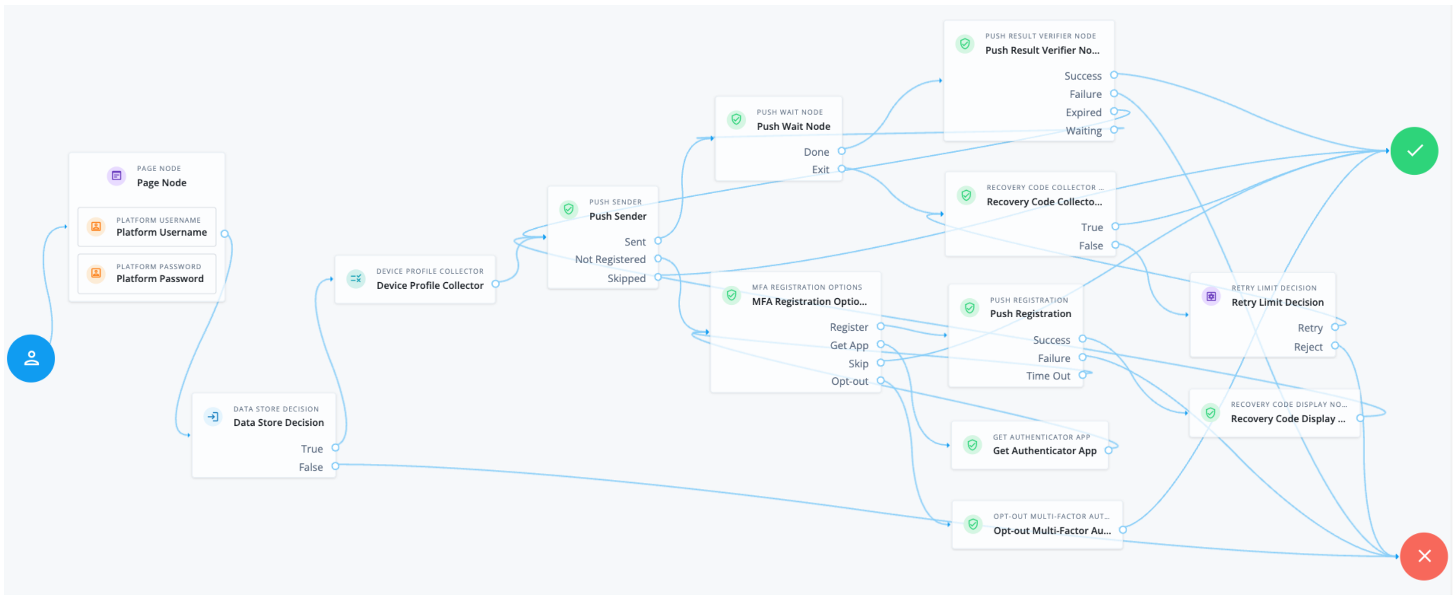
  The node can't identify the user from the shared state.

- `Payload data exceed maximum accepted size`

  The size of the message payload exceeds 3 Kb. Review any custom attributes you've included.

Examples

The following example shows one possible implementation of multi-factor push authentication, which uses this node:



List of node connections

Source node	Outcome path	Target node
Page Node containing nodes to collect credentials. For standalone AM deployments, implement a Username Collector node and a Password Collector node. For Advanced Identity Cloud and Ping Identity Platform deployments, implement a Platform Username node and a Platform Password node.	→	Data Store Decision
Data Store Decision	True	Device Profile Collector
	False	Failure
Device Profile Collector	→	Push Sender
Push Sender	Sent	Push Wait
	Not Registered	MFA Registration Options
	Skipped	Success
Push Wait	Done	Push Result Verifier
	Exit	Recovery Code Collector Decision
Push Result Verifier	Success	Success

Source node	Outcome path	Target node
	Failure	Failure
	Expired	Push Sender
	Waiting	Push Wait
MFA Registration Options	Register	Push Registration
	Get App	Get Authenticator App
	Skip	Success
	Opt-out	Opt-out Multi-Factor Authentication
Recovery Code Collector Decision	True	Success
	False	Retry Limit Decision
Push Registration	Success	Recovery Code Display Node
	Failure	Failure
	Time Out	MFA Registration Options
Get Authenticator App	→	MFA Registration Options
Opt-out Multi-Factor Authentication	→	Success
Retry Limit Decision	Retry	Recovery Code Collector Decision
	Reject	Failure
Recovery Code Display Node	→	Push Sender

After verifying the user's credentials, evaluation continues to the Device Profile Collector node to collect the device's location and then proceeds to the Push Sender node.

If the user *has* a registered device:

1. The Push Sender node sends a push notification to their registered device.
2. The Push Wait node pauses authentication for five seconds. During this time, the user can respond to the push notification on their device using the ForgeRock Authenticator app.

   If the user exits the Push Wait node, they're directed to the Recovery Code Collector Decision node, where they can enter a recovery code to authenticate.

> **Tip**
>
> Configure the **Exit Message** property in the Push Wait node with a message, such as `Lost phone? Use a recovery code` for situations like this.

A Retry Limit Decision node allows three attempts to enter a recovery code before failing the authentication.

3. The Push Result Verifier node verifies the user's response:
    - If the user responds positively, they're authenticated successfully and logged in.
    - If the user responds negatively, authentication fails.
    - If the push notification expires, the Push Sender node sends a new push notification.

    > **Tip**
    >
    > Use a Retry Limit Decision node to constrain the number of times a new code is sent.

    - If the user hasn't yet responded, the flow loops back a step and the Push Wait node pauses authentication for another 5 seconds.

If the user *doesn't have* a registered device:

1. The MFA Registration Options node presents the user with the following options:

    Register Device

    The flow continues to the Push Registration node, which displays a QR code for the user to scan with their authenticator app.

    Get the App

    Displayed only if the node is configured to display Get Authenticator App. The flow continues to the Get Authenticator App node, which displays links to download the authenticator app.

    Skip this step

    Displayed only if the node is configured to allow users to skip registration. In this example, skipping is linked to the `Success` outcome. However, you could provide an alternative authentication flow using an Inner Tree Evaluator node for example.

    Opt-out

    Displayed only if the node is configured to allow users to skip registration. Evaluation continues to the Opt-out Multi-Factor Authentication node, which updates the user's profile to skip MFA with push in the future. In this example, after updating the profile, the flow continues to the `Success` outcome.

2. The user registers the device with the Push Registration node.

    After registration, the Recovery Code Display node displays the recovery codes to the user and the flow returns to the Push Sender node to continue push authentication.

Note

To manage push devices, the user must log in using either the device or a recovery code.
Find more information in the MFA documentation for:

- PingAM
- Advanced Identity Cloud

Push Wait node

The **Push Wait** node pauses authentication for the specified number of seconds during the processing of a push authentication request.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Authenticators

The push-related nodes integrate with the ForgeRock Authenticator app for Android and iOS.

Third-party authenticator apps aren't compatible with the push notification functionality.

Inputs

If the push type in the Push Sender node is set to `Display Challenge Code`, this node checks for the `pushNumberChallengeKey` in the incoming state. This value populates the `{{challenge}}` variable if it's included in the message.

Dependencies

Precede this node in the flow with a Push Sender node to send the push authentication request.

Configuration

Property	Usage
Seconds To Wait	The number of seconds to pause authentication. Default: `5`

Property	Usage
Waiting Message	(Optional) Add a custom, localized message to display to the user. You can use the `{{time}}` variable in the **Value** field to include the remaining seconds in the message: 1. Click **+**. 2. In the **Key** field, enter the locale. For example, `en-gb`.[1] 3. In the **Value** field, enter the message. 4. Click **Done**. 5. Repeat to add more messages and save your changes when you're done. Leave blank to use the default message.[2] Default: `Waiting for response…`
Push Challenge Message	(Optional) Add a custom, localized message to display to the user with the challenge code. You can use the `{{challenge}}` variable in the **Value** field to include the number challenge in the message. This message is displayed only when the **Push Type** in the Push Sender node is set to `Display Challenge Code`: 1. Click **+**. 2. In the **Key** field, enter the locale. For example, `en-gb`.[1] 3. In the **Value** field, enter the message. 4. Click **Done**. 5. Repeat to add more messages and save your changes when you're done. Leave blank to use the default message.[2] Default: `Open your Authenticator app and tap the number shown to sign-in`
Exit Message	(Optional) Add custom, localized text to display on the button the user can click to exit the node before the waiting period has elapsed. 1. Click **+**. 2. In the **Key** field, enter the locale. For example, `en-gb`.[1] 3. In the **Value** field, enter the message. 4. Click **Done**. 5. Repeat to add more messages and save your changes when you're done. Leave blank to use the default message.[2] Default: `Cancel`

[1] Specify a locale that Java supports, such as `en-gb`. Otherwise, the node throws a configuration exception with an `Invalid locale provided` message.

[2] PingAM only: Learn more about customizing and translating default messages in Internationalize nodes.

Outputs

This node doesn't change the shared state.

Outcomes

Done

The waiting time has elapsed.

`Exit`

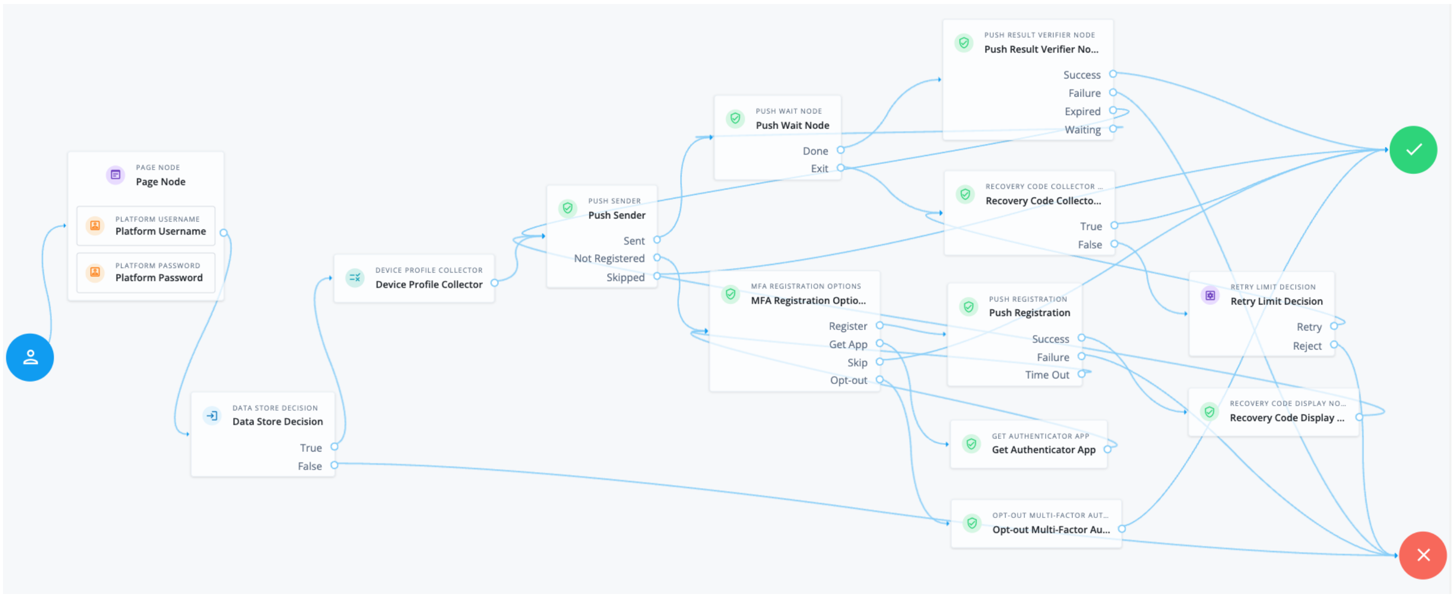
The user clicked the **Cancel** button to exit the node.

Errors

This node doesn't log any error or warning messages of its own.

Examples

The following example shows one possible implementation of multi-factor push authentication, which uses this node:



List of node connections

Source node	Outcome path	Target node
Page Node containing nodes to collect credentials. For standalone AM deployments, implement a Username Collector node and a Password Collector node. For Advanced Identity Cloud and Ping Identity Platform deployments, implement a Platform Username node and a Platform Password node.	→	Data Store Decision

Source node	Outcome path	Target node
Data Store Decision	True	Device Profile Collector
	False	Failure
Device Profile Collector	→	Push Sender
Push Sender	Sent	Push Wait
	Not Registered	MFA Registration Options
	Skipped	Success
Push Wait	Done	Push Result Verifier
	Exit	Recovery Code Collector Decision
Push Result Verifier	Success	Success
	Failure	Failure
	Expired	Push Sender
	Waiting	Push Wait
MFA Registration Options	Register	Push Registration
	Get App	Get Authenticator App
	Skip	Success
	Opt-out	Opt-out Multi-Factor Authentication
Recovery Code Collector Decision	True	Success
	False	Retry Limit Decision
Push Registration	Success	Recovery Code Display Node
	Failure	Failure
	Time Out	MFA Registration Options
Get Authenticator App	→	MFA Registration Options
Opt-out Multi-Factor Authentication	→	Success
Retry Limit Decision	Retry	Recovery Code Collector Decision

Source node	Outcome path	Target node
	Reject	Failure
Recovery Code Display Node	→	Push Sender

After verifying the user's credentials, evaluation continues to the Device Profile Collector node to collect the device's location and then proceeds to the Push Sender node.

If the user *has* a registered device:

1. The Push Sender node sends a push notification to their registered device.
2. The Push Wait node pauses authentication for five seconds. During this time, the user can respond to the push notification on their device using the ForgeRock Authenticator app.

   If the user exits the Push Wait node, they're directed to the Recovery Code Collector Decision node, where they can enter a recovery code to authenticate.

   > **Tip**
   >
   > Configure the **Exit Message** property in the Push Wait node with a message, such as `Lost phone? Use a recovery code` for situations like this.

   A Retry Limit Decision node allows three attempts to enter a recovery code before failing the authentication.
3. The Push Result Verifier node verifies the user's response:
   - If the user responds positively, they're authenticated successfully and logged in.
   - If the user responds negatively, authentication fails.
   - If the push notification expires, the Push Sender node sends a new push notification.

     > **Tip**
     >
     > Use a Retry Limit Decision node to constrain the number of times a new code is sent.
   - If the user hasn't yet responded, the flow loops back a step and the Push Wait node pauses authentication for another 5 seconds.

If the user *doesn't have* a registered device:

1. The MFA Registration Options node presents the user with the following options:

   Register Device

   The flow continues to the Push Registration node, which displays a QR code for the user to scan with their authenticator app.

   Get the App

   Displayed only if the node is configured to display Get Authenticator App. The flow continues to the Get Authenticator App node, which displays links to download the authenticator app.

Skip this step

Displayed only if the node is configured to allow users to skip registration. In this example, skipping is linked to the `Success` outcome. However, you could provide an alternative authentication flow using an Inner Tree Evaluator node for example.

Opt-out

Displayed only if the node is configured to allow users to skip registration. Evaluation continues to the Opt-out Multi-Factor Authentication node, which updates the user's profile to skip MFA with push in the future. In this example, after updating the profile, the flow continues to the `Success` outcome.

2. The user registers the device with the Push Registration node.

   After registration, the Recovery Code Display node displays the recovery codes to the user and the flow returns to the Push Sender node to continue push authentication.

Note

To manage push devices, the user must log in using either the device or a recovery code.
Find more information in the MFA documentation for:

- PingAM
- Advanced Identity Cloud

Recovery Code Collector Decision node

The **Recovery Code Collector Decision** node lets users authenticate with a recovery code that was generated when they registered a device for multi-factor authentication (MFA).

Use this node in an authentication journey that includes push notifications or one-time passwords. When the user loses their registered device, they can use a recovery code as an alternative method for authentication.

Find more information on viewing recovery codes when registering a device in the *ForgeRock Authenticator* documentation for:

- PingAM
- Advanced Identity Cloud

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node requires the `realm` and `username` properties in the incoming node state.

Implement a Username Collector node (standalone AM) or Platform Username node (Advanced Identity Cloud and Ping Identity Platform deployments) earlier in the journey.

Dependencies

This node has no dependencies.

Configuration

Property	Usage
Recovery Code Type	Select the type of recovery code the user will submit for verification. Default: `OATH`

Outputs

This node doesn't change the shared state.

Outcomes

`True`

The user entered a valid recovery code.

`False`

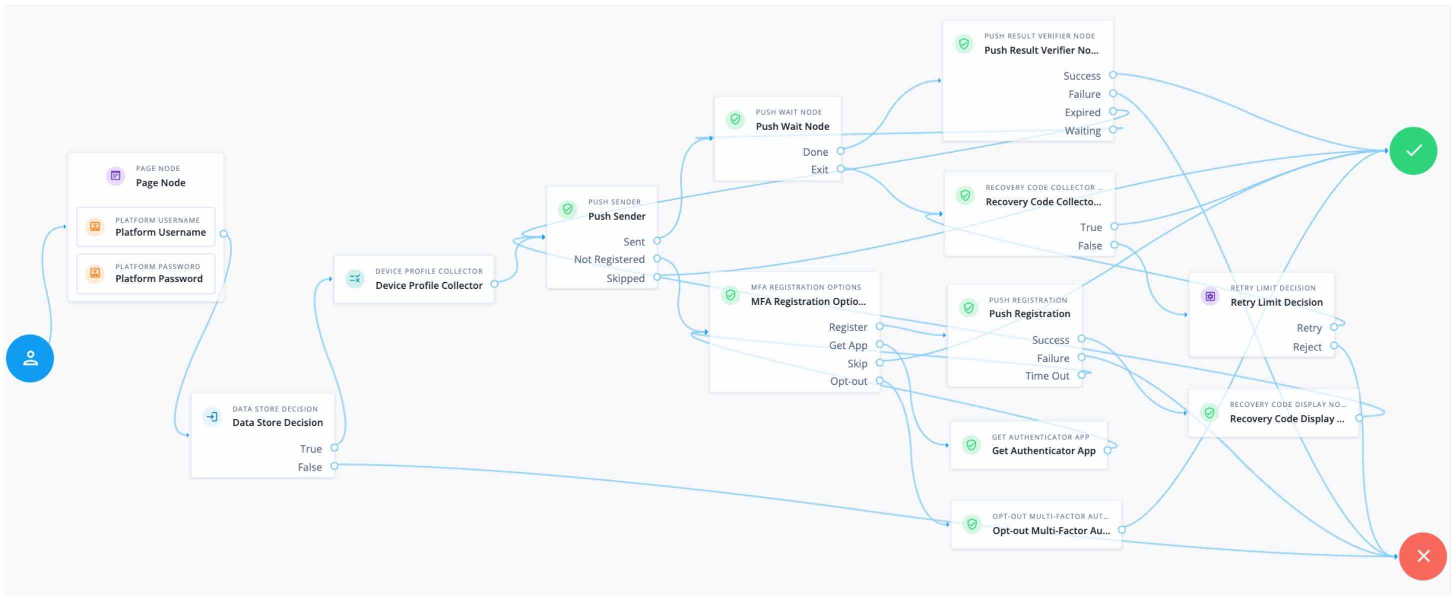
The user didn't enter a valid recovery code.

Errors

This node doesn't log any error or warning messages of its own.

Examples

The following example shows one possible implementation of multi-factor push authentication, which uses this node:



List of node connections

Source node	Outcome path	Target node
Page Node containing nodes to collect credentials. For standalone AM deployments, implement a Username Collector node and a Password Collector node. For Advanced Identity Cloud and Ping Identity Platform deployments, implement a Platform Username node and a Platform Password node.	→	Data Store Decision
Data Store Decision	True	Device Profile Collector
	False	Failure
Device Profile Collector	→	Push Sender
Push Sender	Sent	Push Wait
	Not Registered	MFA Registration Options
	Skipped	Success
Push Wait	Done	Push Result Verifier
	Exit	Recovery Code Collector Decision
Push Result Verifier	Success	Success

Source node	Outcome path	Target node
	Failure	Failure
	Expired	Push Sender
	Waiting	Push Wait
MFA Registration Options	Register	Push Registration
	Get App	Get Authenticator App
	Skip	Success
	Opt-out	Opt-out Multi-Factor Authentication
Recovery Code Collector Decision	True	Success
	False	Retry Limit Decision
Push Registration	Success	Recovery Code Display Node
	Failure	Failure
	Time Out	MFA Registration Options
Get Authenticator App	→	MFA Registration Options
Opt-out Multi-Factor Authentication	→	Success
Retry Limit Decision	Retry	Recovery Code Collector Decision
	Reject	Failure
Recovery Code Display Node	→	Push Sender

After verifying the user's credentials, evaluation continues to the Device Profile Collector node to collect the device's location and then proceeds to the Push Sender node.

If the user *has* a registered device:

1. The Push Sender node sends a push notification to their registered device.
2. The Push Wait node pauses authentication for five seconds. During this time, the user can respond to the push notification on their device using the ForgeRock Authenticator app.

   If the user exits the Push Wait node, they're directed to the Recovery Code Collector Decision node, where they can enter a recovery code to authenticate.

> **Tip**
>
> Configure the **Exit Message** property in the Push Wait node with a message, such as `Lost phone? Use a recovery code` for situations like this.

A Retry Limit Decision node allows three attempts to enter a recovery code before failing the authentication.

3. The Push Result Verifier node verifies the user's response:
   - If the user responds positively, they're authenticated successfully and logged in.
   - If the user responds negatively, authentication fails.
   - If the push notification expires, the Push Sender node sends a new push notification.

     > **Tip**
     >
     > Use a Retry Limit Decision node to constrain the number of times a new code is sent.

   - If the user hasn't yet responded, the flow loops back a step and the Push Wait node pauses authentication for another 5 seconds.

If the user *doesn't have* a registered device:

1. The MFA Registration Options node presents the user with the following options:

   Register Device

   The flow continues to the Push Registration node, which displays a QR code for the user to scan with their authenticator app.

   Get the App

   Displayed only if the node is configured to display Get Authenticator App. The flow continues to the Get Authenticator App node, which displays links to download the authenticator app.

   Skip this step

   Displayed only if the node is configured to allow users to skip registration. In this example, skipping is linked to the `Success` outcome. However, you could provide an alternative authentication flow using an Inner Tree Evaluator node for example.

   Opt-out

   Displayed only if the node is configured to allow users to skip registration. Evaluation continues to the Opt-out Multi-Factor Authentication node, which updates the user's profile to skip MFA with push in the future. In this example, after updating the profile, the flow continues to the `Success` outcome.

2. The user registers the device with the Push Registration node.

   After registration, the Recovery Code Display node displays the recovery codes to the user and the flow returns to the Push Sender node to continue push authentication.

Note

To manage push devices, the user must log in using either the device or a recovery code.
Find more information in the MFA documentation for:

- PingAM
- Advanced Identity Cloud

Recovery Code Display node

The **Recovery Code Display** node retrieves the generated recovery codes from the transient state and displays them to the user, for safe-keeping. The codes can be used to authenticate if a registered device is lost or stolen.

Important

The generated recovery codes can't be retrieved from the user's profile because they're one-way encrypted.
This node provides the *only* opportunity to view and save the recovery codes.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node requires the `recoveryCodes` attribute in the incoming state so that it can display the recovery codes to the user. If there aren't any recovery codes in the transient state, evaluation continues along the only outcome path and nothing is displayed to the user.

Implement an OATH Registration node, a Push Registration node or a WebAuthn Registration node earlier in the journey, and connect the `Success` outcome path to this node to display the codes.

Dependencies

This node has no dependencies.

Configuration

This node has no configurable properties.

Outputs

This node removes the recovery code details from the shared state.

Outcomes

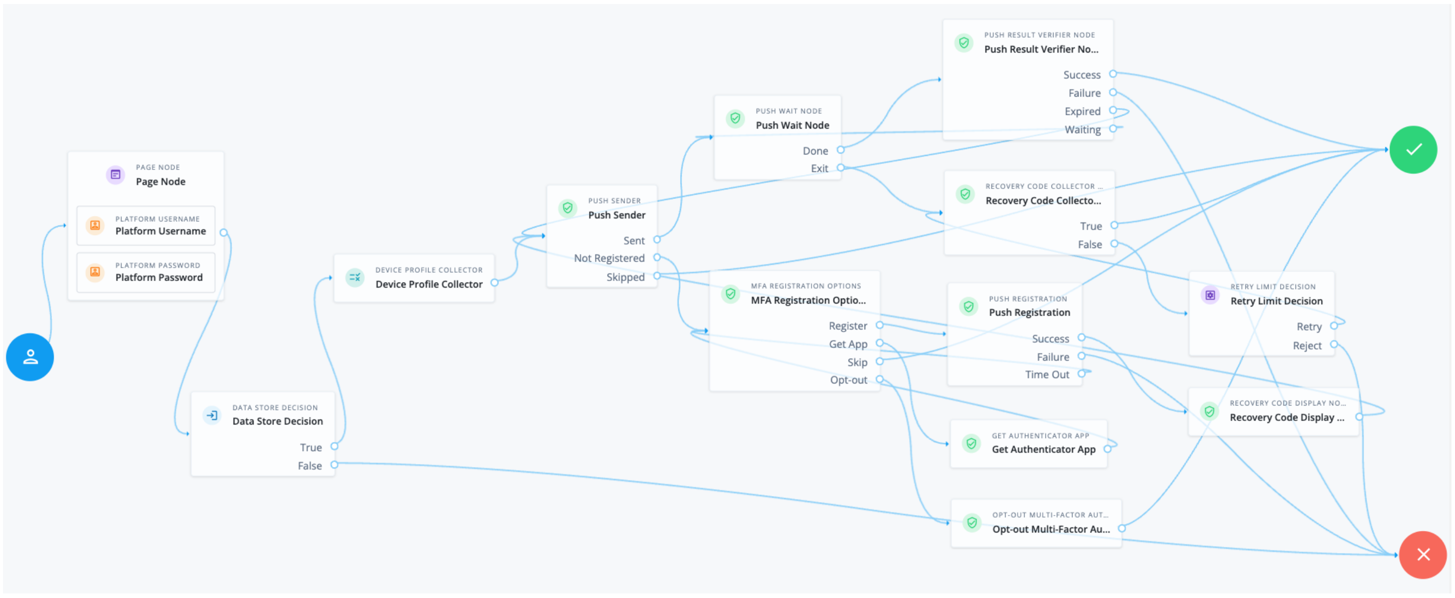
Single outcome path.

Errors

This node doesn't log any error or warning messages of its own.

Examples

The following example shows one possible implementation of multi-factor push authentication, which uses this node:



List of node connections

Source node	Outcome path	Target node
Page Node containing nodes to collect credentials. For standalone AM deployments, implement a Username Collector node and a Password Collector node. For Advanced Identity Cloud and Ping Identity Platform deployments, implement a Platform Username node and a Platform Password node.	→	Data Store Decision
Data Store Decision	True	Device Profile Collector

Source node	Outcome path	Target node
	False	Failure
Device Profile Collector	→	Push Sender
Push Sender	Sent	Push Wait
	Not Registered	MFA Registration Options
	Skipped	Success
Push Wait	Done	Push Result Verifier
	Exit	Recovery Code Collector Decision
Push Result Verifier	Success	Success
	Failure	Failure
	Expired	Push Sender
	Waiting	Push Wait
MFA Registration Options	Register	Push Registration
	Get App	Get Authenticator App
	Skip	Success
	Opt-out	Opt-out Multi-Factor Authentication
Recovery Code Collector Decision	True	Success
	False	Retry Limit Decision
Push Registration	Success	Recovery Code Display Node
	Failure	Failure
	Time Out	MFA Registration Options
Get Authenticator App	→	MFA Registration Options
Opt-out Multi-Factor Authentication	→	Success
Retry Limit Decision	Retry	Recovery Code Collector Decision
	Reject	Failure

Source node	Outcome path	Target node
Recovery Code Display Node	→	Push Sender

After verifying the user's credentials, evaluation continues to the Device Profile Collector node to collect the device's location and then proceeds to the Push Sender node.

If the user *has* a registered device:

1. The Push Sender node sends a push notification to their registered device.
2. The Push Wait node pauses authentication for five seconds. During this time, the user can respond to the push notification on their device using the ForgeRock Authenticator app.

   If the user exits the Push Wait node, they're directed to the Recovery Code Collector Decision node, where they can enter a recovery code to authenticate.

   > **Tip**
   >
   > Configure the **Exit Message** property in the Push Wait node with a message, such as `Lost phone? Use a recovery code` for situations like this.

   A Retry Limit Decision node allows three attempts to enter a recovery code before failing the authentication.
3. The Push Result Verifier node verifies the user's response:
   - If the user responds positively, they're authenticated successfully and logged in.
   - If the user responds negatively, authentication fails.
   - If the push notification expires, the Push Sender node sends a new push notification.

     > **Tip**
     >
     > Use a Retry Limit Decision node to constrain the number of times a new code is sent.
   - If the user hasn't yet responded, the flow loops back a step and the Push Wait node pauses authentication for another 5 seconds.

If the user *doesn't have* a registered device:

1. The MFA Registration Options node presents the user with the following options:

   Register Device

   The flow continues to the Push Registration node, which displays a QR code for the user to scan with their authenticator app.

   Get the App

   Displayed only if the node is configured to display Get Authenticator App. The flow continues to the Get Authenticator App node, which displays links to download the authenticator app.

Skip this step

Displayed only if the node is configured to allow users to skip registration. In this example, skipping is linked to the `Success` outcome. However, you could provide an alternative authentication flow using an Inner Tree Evaluator node for example.

Opt-out

Displayed only if the node is configured to allow users to skip registration. Evaluation continues to the Opt-out Multi-Factor Authentication node, which updates the user's profile to skip MFA with push in the future. In this example, after updating the profile, the flow continues to the `Success` outcome.

2. The user registers the device with the Push Registration node.

   After registration, the Recovery Code Display node displays the recovery codes to the user and the flow returns to the Push Sender node to continue push authentication.

Note

To manage push devices, the user must log in using either the device or a recovery code.
Find more information in the MFA documentation for:

- PingAM
- Advanced Identity Cloud

WebAuthn Authentication node

The **WebAuthn Authentication** node lets users on supported clients authenticate using a registered FIDO device.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

The node reads the `username` from the shared state. Implement the following node before this node in the journey:

- Username Collector node (standalone AM)
- Platform Username node (Ping Identity Platform deployment)

Prerequisites

For successful authentication, this node depends on:

- A client that supports web authentication
- A registered FIDO device

Configuration

Property	Usage
Relying party identifier	The domain used as the relying party identifier during web authentication. This is the domain against which to register the device. If you leave this field blank, it defaults to the domain name of the AM instance, for example, `am.example.com`. Specify an alternative domain if your AM instances are behind a load balancer, for example.
Origin domains	A list of fully qualified URLs to accept as the origin of the incoming request. If this field is empty, the accepted origin is the incoming request origin.
User verification requirement	The required user verification level. The available options are: **`REQUIRED`** The authenticator used must verify the user's identity, for example, by using biometrics. Authenticators that don't verify the user's identity are filtered out and can't be selected by the user. **`PREFERRED`** If multiple authenticators are presented, AM prefers those that verify the user's identity. If none are available, AM accepts any authenticator. **`DISCOURAGED`** AM doesn't require an authenticator that verifies the user's identity. Authenticators that don't verify the user's identity are preferred.
Allow recovery codes	If you select this option, AM lets the user enter a recovery code instead of performing an authentication gesture. Enabling this options adds a `Recovery Code` outcome path to the node. The outcome path should lead to a Recovery Code Collector Decision node to collect and verify the recovery code.
Timeout	The number of seconds to wait for a valid WebAuthn authenticator to be registered before failing. If the specified timeout is reached, evaluation continues along the `Client error` outcome path. AM stores a message in the `WebAuthenticationDOMException` property of the shared state.

Property	Usage
Username from device	Specifies whether AM should get the username from the device. If you enable this option and the device is unable to store or provide usernames, the node fails and evaluation continues along the `Failure` path. Find information on using this property for usernameless authentication with ForgeRock Go in Configure usernameless authentication with ForgeRock Go.
Return challenge as JavaScript	If you enable this option, the node returns its challenge as a fully encapsulated client-side JavaScript that interacts directly with the WebAuthn API and submit the response back. If this option is disabled, the node returns the challenge and associated data in a metadata callback. A custom UI, for example an application using the Ping SDKs, uses the information from the callback to interact with the WebAuthn API on AM's behalf.

Outcomes

`Unsupported`

If the user's client doesn't support web authentication, evaluation continues along the `Unsupported` outcome path. For example, clients connected over the HTTP protocol rather than HTTPS don't support WebAuthn; however, HTTPS may not be required when testing locally on `http://localhost`. For more information, refer to Is origin potentially trustworthy?.

`No Device Registered`

If the user doesn't have a registered device, evaluation continues along the `No Device Registered` outcome path.

`Success`

If the user successfully authenticates with a device of the type determined by the **User verification requirement** property, evaluation continues along the `Success` outcome path.

`Failure`

If the node encounters an issue when attempting to authenticate the user with the device, evaluation continues along the `Failure` outcome path; for example, if the node can't verify that the response from the authenticator was appropriate for the specific instance of the authentication journey.

`Client Error`

If the user's client encounters an issue when attempting to authenticate using the device, for example, if the timeout was reached, evaluation continues along the `Client Error` outcome path.

The journey takes this path whenever the client throws a `DOMException`, as required by the Web Authentication: An API for accessing Public Key Credentials Level 1 specification.

`Recovery Code`

If **Allow recovery code** is enabled, the node gives the user an option to enter a recovery code rather than authenticate using a device. If the user enters a recovery code, evaluation continues along the `Recovery Code` outcome path.

This outcome path must lead to a Recovery Code Collector Decision node to let AM accept and verify the recovery code.

Outputs

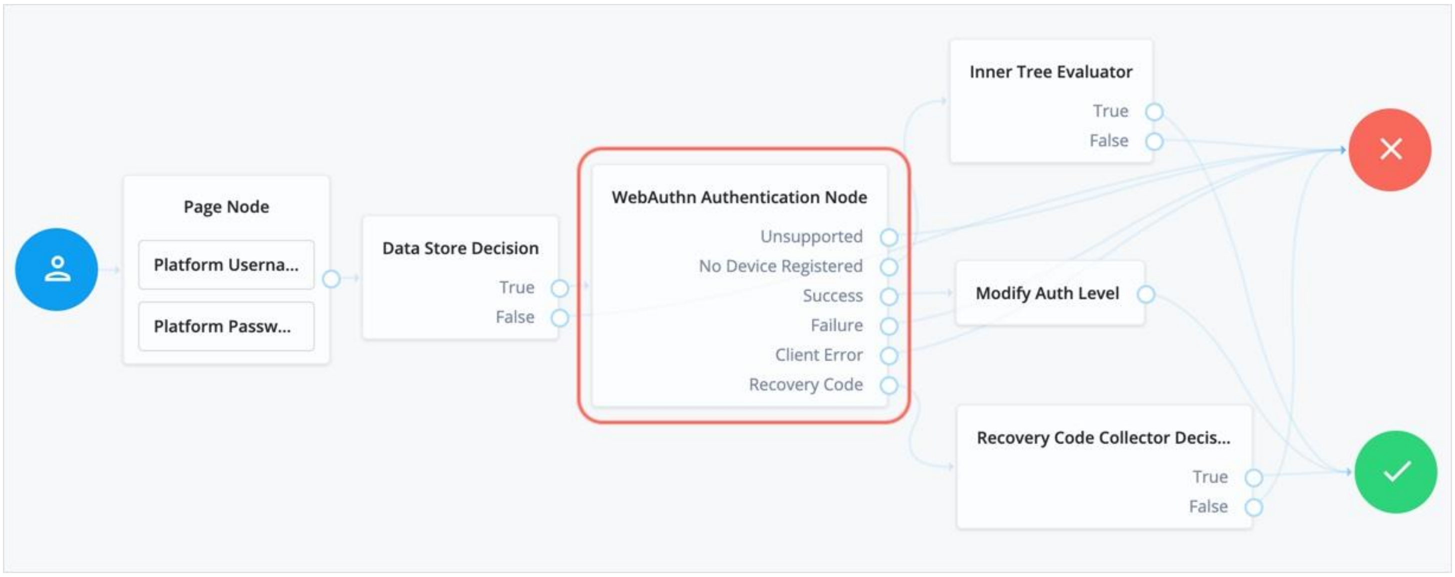
If a client error occurs, the node adds the error type and description to a property named `WebAuthenticationDOMException` in the shared state. Other nodes can read this property later in the journey, if required.

Note

The contents of the transient state for this node aren't public. Don't rely on them in your scripts.

Example

This example shows one possible implementation of the flow for authenticating with WebAuthn devices:

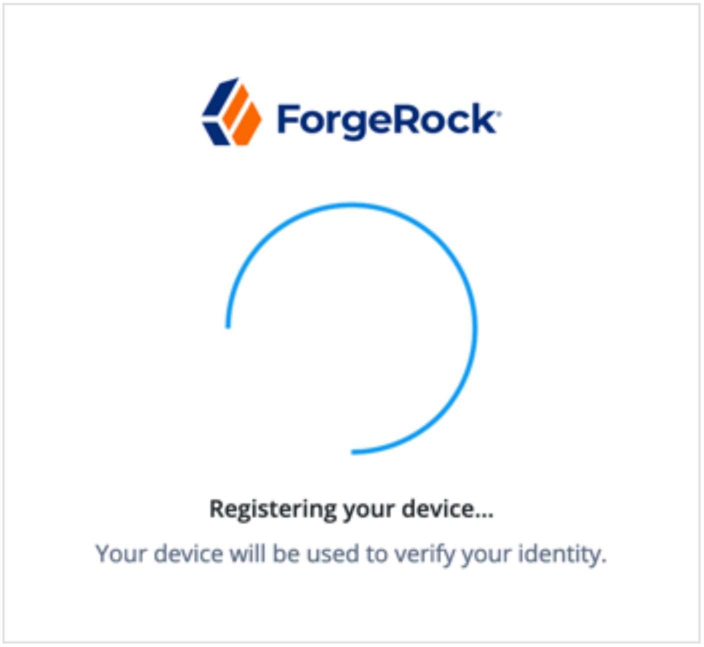


After verifying the users credentials against the configured data store, evaluation continues to the WebAuthn Authentication node.

If the user's client doesn't support WebAuthn, authentication fails and the user doesn't get a session. A more user-friendly approach would be to set a success URL to redirect the user to a page explaining the benefits of multi-factor authentication, and then proceeding to the `Success` node.

If there are no registered WebAuthn devices present in the user's profile, the failure URL is set, pointing to a flow that lets the user register a device. This stage could also be an Inner Tree Evaluator node.

If the user's client supports WebAuthn, and the connection is secured with TLS, the user is prompted to complete an authorization gesture, for example, scanning a fingerprint, or entering a PIN:



Registering your device...

Your device will be used to verify your identity.

The user's browser may present a consent pop-up to allow access to the authenticators available on the client. When consent has been granted, the browser activates the relevant authenticators, ready for authentication.

> **Tip**
>
> The relying party details configured in the node are often included in the consent message to help the user verify the entity requesting access.

The authenticators the client activates for authentication depend on the value of the properties in the node. For example, if the **User verification requirement** property is set to `REQUIRED`, the client **SHOULD** only activate authenticators that verify the identity of the user.

For extra protection, AM **WILL** verify that the response from an authenticator matches the criteria configured for the node, and will reject an authentication attempt by an inappropriate authenticator type by routing it to the `Failure` outcome.

When the user completes an authorization gesture, for example, by scanning a fingerprint or entering a PIN, evaluation continues along the `Success` outcome path. In this example, their authentication level is increased by ten to signify the stronger authentication that has occurred, and the user is taken to their profile page.

If the user clicks the `Use Recovery Code` button, evaluation continues to the Recovery Code Collector Decision node, ready to accept the recovery code. If verified, the user is taken to their profile page.

Any problems encountered during authentication lead to the `Failure` outcome, including a timeout, or to the `Client Error` outcome, resulting in an authentication failure.

WebAuthn Device Storage node

Writes information about FIDO2 devices to a user's profile. The user can subsequently authenticate using the device.

Use this node to store the device data the WebAuthn Registration node places into the transient node state when its **Store device data in transient state** property is enabled.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `Success`
- `Failure`
- `Exceed Device Limit`

If AM encounters an issue when attempting to save the device data to the user's profile; for example, the user was not identified earlier, then evaluation continues along the `Failure` outcome path.

If the **Maximum Saved Devices** property is set to an integer greater than zero, and registering a new device would take the number of devices above the specified threshold, then evaluation continues down the `Exceed Device Limit` outcome path. In this case, you may need to instruct your users to log in with an existing device in order to remove one or more of their registered devices.

If the node successfully stores the device data to the user's profile, evaluation continues along the `Success` outcome path.

Properties

Property	Usage
Generate recovery codes	Specify whether WebAuthn device recovery codes should be generated. If enabled, recovery codes are generated and stored in the transient node state, and stored alongside the device profile. Use the Recovery Code Display node to display the codes to the user for safe keeping. **Important** Generating recovery codes overwrites all existing WebAuthn device recovery codes for the device. Only the most recent set of recovery codes can be used for authentication if a device has been lost or stolen.
Maximum Saved Devices	Specify the maximum number of WebAuthn devices to save in a user's profile. Set this property to `0` if you do not want to limit the number of devices. When this property is greater than zero, the `Exceed Device Limit` outcome path becomes available.

WebAuthn Registration node

Lets users of supported clients register FIDO2 devices for use during authentication.

AM interacts with FIDO2/WebAuthn capable browsers, such as `Chrome`, `Firefox` and `Microsoft Edge`. These browsers interact with CTAP2 authenticators, including U2F and FIDO2 Security Keys, and platforms, such as Windows Hello or Apple Touch ID.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `Unsupported`
- `Success`
- `Failure`
- `Client Error`
- `Exceed Device Limit`

If the user's client does not support WebAuthn, evaluation continues along the `Unsupported` outcome path. For example, clients connected over the HTTP protocol rather than HTTPS do not support WebAuthn.

If AM encounters an issue when attempting to register using a device, evaluation continues along the `Failure` outcome path. For example, AM could not verify the response from the authenticator was appropriate for the specific instance of the authentication ceremony.

If the user's client encounters an issue when attempting to register using a device, for example, if the timeout was reached, then evaluation continues along the `Client Error` outcome path. This outcome is used whenever the client throws a `DOMException`, as required by the Web Authentication: An API for accessing Public Key Credentials Level 1 specification.

Tip

If a client error occurs, the error type and description are added to a property named `WebAuthenticationDOMException` in the shared state. This property can be read by other nodes later, if required.

If the **Maximum Saved Devices** property is set to an integer greater than zero, and registering a new device would take the number of devices above the specified threshold, then evaluation continues down the `Exceed Device Limit` outcome path. In this case, you may need to instruct your users to log in with an existing device in order to remove one or more of their registered devices.

If the user successfully registers an authenticator of the correct type as determined by the node's properties, evaluation continues along the `Success` outcome path.

Properties

Property	Usage
Relying party	Specify the name of the relying party entity registering and authenticating users by using WebAuthn. For example, `Example Inc.`
Relying party identifier	Specifies the domain used as the relying party identifier during WebAuthn. If not specified, AM uses the domain name of the instance, such as `am.example.com`. Specify an alternative domain if your AM instances are behind a load balancer, for example.
Origin domains	Specifies a list of fully qualified URLs to accept as the origin of incoming requests. If left empty, AM accepts any incoming domain.
User verification requirement	Specifies the required level of user verification. The available options are: **`REQUIRED`** The authenticator used must verify the identity of the user, for example by using biometrics. Authenticators that do not verify the identity of the user should not be activated for registration. **`PREFERRED`** Use of an authenticator that verifies the identity of the user is preferred, but if none are available any authenticator is accepted. **`DISCOURAGED`** Use of an authenticator that verifies the identity of the user is not required. Authenticators that do not verify the identity of the user should be preferred.

Property	Usage
Preferred mode of attestation	Specifies whether AM requires that the authenticator provides attestation statements. The available options are: **`NONE`** AM does not require the authenticator to provide attestation statements. If the authenticator does send attestation statements, AM *will not* verify them, and will not fail the process. **`INDIRECT`** AM does not require the authenticator to provide attestation statements. If the authenticator does send attestation statements, AM *will* verify them, and will fail the process if they fail verification. **`DIRECT`** AM requires the authenticator provides attestation statements, and *will* verify them. The process will fail if the attestation statements cannot be verified. AM supports the following attestation formats: • None • Android SafetyNet • Packed • FIDO U2F • TPM **Important** You must set the **Preferred mode of attestation** property to `NONE` to use an authenticator that provides attestation statements in a format other than the supported formats above. Specifically, AM *does not* currently support: • android-safetynet • android-key
Accepted signing algorithms	Specify the algorithms authenticators can use to sign their assertions.

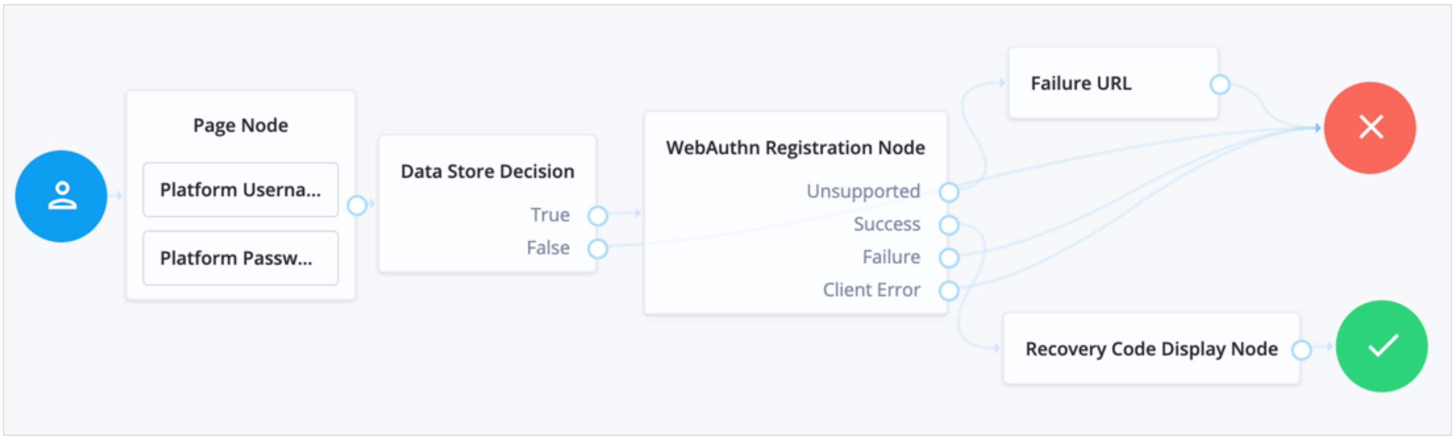
Property	Usage
Authentication attachment	Specifies whether AM requires that the authenticator is a particular attachment type. There are two types of authenticator attachments: • An authenticator that is built-in to the client device is labeled a *platform attachment*. A fingerprint scanner built-in to a phone or laptop is an example of a platform attachment authenticator. • An authenticator that can roam, or move, between different client devices is labeled a *cross-platform attachment*. A USB hardware security key is an example of a cross-platform attachment authenticator. The available options are: **`UNSPECIFIED`** AM accepts any attachment type. **`PLATFORM`** The authenticator must be a *platform* attachment type. The client should not activate other authenticator types for registration. **`CROSS_PLATFORM`** The authenticator must be a *cross-platform* attachment type. The client should not activate other authenticator types for registration.
Trust Store alias	Specifies the name of a secret store configured in the realm that contains CA-issued certificate chains, which can be used to verify attestation data provided by a device. The alias of the realm trust store holding the secrets necessary to validate a supplied attestation certificate. The alias name must only contain the characters `a-z` and the `.` symbol. The value is also appended to the string `am.authentication.nodes.webauthn.truststore.` to form the dynamic secret ID used to map the certificate chains.
Enforce revocation check	Specifies whether to enforce certificate revocation checks. When enabled, then any attestation certificate's trust chain *MUST* have a CRL or OCSP entry that can be verified by AM during processing. When disabled, certificates are not checked for revocation. You must ensure expired or revoked certificates are manually removed.
Timeout	Specify the number of seconds to wait for a response from an authenticator. If the specified time is reached, evaluation continues along the `Client error` outcome path, and a relevant message is stored in the `WebAuthenticationDOMException` property of the shared state.

Property	Usage
Limit registrations	Specify whether the same authenticator can be registered multiple times. If enabled, the client should not activate an authenticator that is already registered for registration.
Generate recovery codes	Specify whether WebAuthn-specific recovery codes should be generated. If enabled, recovery codes are generated and stored in transient state if registration was successful. Use the Recovery Code Display node to display the codes to the user for safe-keeping. Don't enable this property if you've enabled the **Store device data in transient state** property (and aren't saving the device data to the user's profile immediately). Enable the **Generate recovery codes** property in the WebAuthn Device Storage node instead. **Important** Generating recovery codes will overwrite all existing WebAuthn-specific recovery codes. Only the most recent set of recovery codes can be used for authentication if a device has been lost or stolen.
Store data in transient state	Specify whether the information provided by the device to the node is stored in the transient node state for later analysis by subsequent nodes, using the key `webauthnData`. In addition to the information provided by the device, the type of attestation achieved; for example, `BASIC`, `CA`, `SELF` and so on, is stored in the transient node state, using the key `webauthnAttestationType`. **Warning** The amount of data involved can be large. Only enable this option if you intend to analyze it.
Store device data in transient state	Specify whether the information about the device required for WebAuthn is stored in the transient node state rather than saved immediately to the user's profile. Enable this option if you intend to make decisions in scripts, and have enabled the **Store data in transient state** property, and therefore do not want to register the device to the user until the outcome of the analysis is complete. **Important** Do not alter the data while it is in the transient node state, nor when saved to a user's profile. Modifying the device data will likely cause the device to be unable to authenticate. Use the WebAuthn Device Storage node to write the device data to the user's profile when this option is enabled. When disabled, device data is written automatically to the user's profile when registration is successful.

Property	Usage
Username to device	Specifies whether AM requests that the device stores the user's username. When enabled, if the device is unable to store or provide usernames, the node will fail and results in the *Failure* outcome. Learn more about using this property for usernameless authentication with ForgeRock Go in Configure usernameless authentication with ForgeRock Go.
Shared state attribute for display name	Specifies a variable in shared node state that contains a display name for the user; for example, their full name, or email address. The value is written to devices alongside the username when the **Username to device** property is enabled, and helps the user select between the accounts they may have on their devices. If not specified, or the variable is not found in shared state, the username is used. Learn more about using this property for usernameless authentication with ForgeRock Go in Configure usernameless authentication with ForgeRock Go.
Return challenge as JavaScript	Specifies that the node returns its challenge as a fully encapsulated client-side JavaScript that interacts directly with the WebAuthn API, and auto-submits the response back. If disabled, the node returns the challenge and associated data in a metadata callback. A custom UI, for example, an application using the Ping SDKs, uses the information from the callback to interact with the WebAuthn API on AM's behalf.
Maximum Saved Devices	Specifies the maximum number of WebAuthn devices stored in the user's profile. Set this property to `0` if you do not want to limit the number of devices. When this property is greater than zero, the `Exceed Device Limit` outcome path becomes available. **Important** You can only limit the number of devices stored in the user's profile. If the **Store device data in transient state** property is enabled then the node is unable to limit the number of devices, and the `Exceed Device Limit` outcome path is not displayed. In this case, specify the maximum number of saved devices in the WebAuthn Device Storage node.

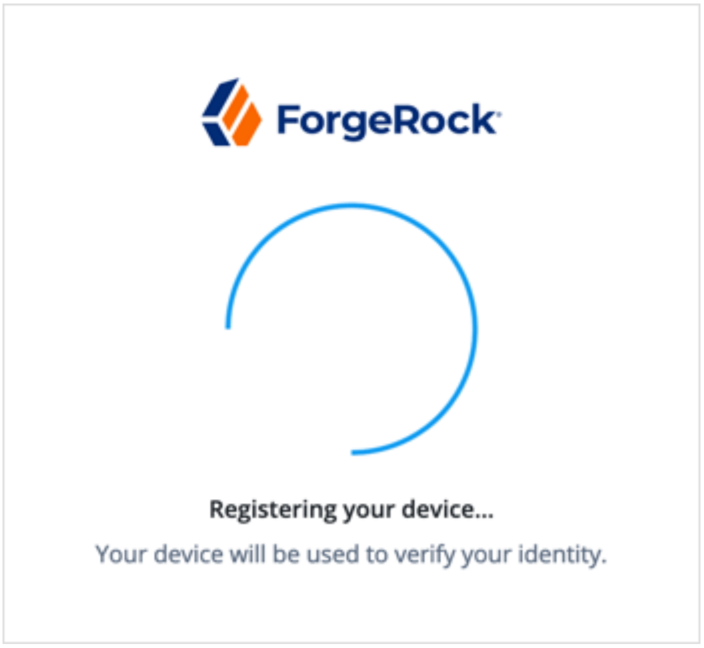
Example

The following example registers WebAuthn devices:



If the user’s client does not support WebAuthn, the failure URL is altered, for example to redirect the user to a page explaining which clients and operating systems support WebAuthn.

If the user’s client does support WebAuthn, and the connection is secured with TLS, AM prompts the user to register an authenticator:



ForgeRock

Registering your device...

Your device will be used to verify your identity.

The user’s browser may present a consent pop-up to allow access to the authenticators available on the client. When consent has been granted, the browser activates the relevant authenticators, ready for registration.

Tip

The relying party details configured in the node are often included in the consent message to help the user verify the entity requesting access.

The authenticators the client activates for registration depend on the value of the properties in the node. For example, if the **User verification requirement** property is set to `REQUIRED`, the client would not activate a USB hardware security key for registration.

When the user completes an authorization gesture, for example, by scanning a fingerprint or entering a PIN, the evaluation continues along the `Success` outcome path, and in this example will be taken to their profile page.

The registered authenticator appears on the user’s dashboard page, with the label *New Security Key*. To rename the authenticator, click its vertical ellipsis context icon, ⋮, and click Rename.

Any problems encountered during the registration, including a timeout, results in the evaluation continuing to the `Failure` outcome.

Risk management nodes

Account Active Decision node

The **Account Active Decision** node determines whether the current account is both active and unlocked, and lets the journey make a decision, based on that check.

An account is considered locked under these conditions:

- The status is inactive.
- The status is active and a duration lockout is set on the account.

An account is considered unlocked under this condition:

- The status is active and no duration lockout is set on the account.

The node determines whether the account has been locked through both persistent (physical) lockout and duration lockout. Learn more in Account lockout for trees.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

The node reads the `username` from the shared state. Implement the following node before this node in the journey:

- Username Collector node (standalone AM)
- Platform Username node (Ping Identity Platform deployment)

Dependencies

This node has no dependencies.

Configuration

This node has no configurable properties.

Outputs

This node doesn't write anything to the shared state.

Outcomes

- `True`

  The journey follows this outcome path if the account is assessed to be `active` and `unlocked`.

- `False`

  The journey follows this outcome path if the account is assessed to be `inactive` or `locked`.

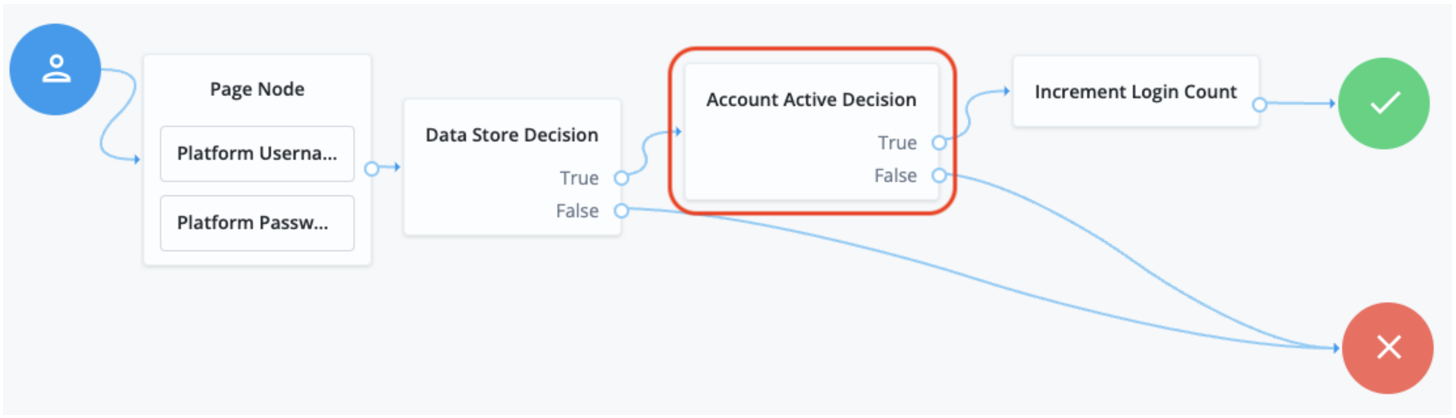
Errors

If the node cannot read the identity of the account, it throws the following exception:

`Failed to get the identity object`

Examples

In this simple login journey, authentication fails if the account is assessed to be `inactive` or `locked`.



This example uses the following nodes:

- The Page node prompts the user to input their username and password:
  - The Platform Username node collects the username and stores it in the shared state.
  - The Platform Password node collects the password and stores it in the shared state.
- The Data Store Decision node uses the username and password to determine whether the account exists.
- The **Account Active Decision** node determines whether the account is active and unlocked.

- If the account is active and unlocked, the Increment Login Count node increments the login count and authentication succeeds.
- If the account is inactive or locked, the authentication fails.

Account Lockout node

The **Account Lockout** node locks or unlocks the authenticating user's account profile.

The node also determines whether the account has been locked through both persistent (physical) lockout and duration lockout. For more information, refer to Account lockout for trees.

Tip

You can also use the Account Active Decision node to check whether the account is locked at any point in the journey.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node requires the `username` property in the incoming node state. It uses this information to access the account status in the user profile.

It also requires the `realm` property, which AM sets by default.

Dependencies

This node depends on the underlying identity service that stores the user profile.

Configuration

Property	Usage
Lock Action	Choose whether to `LOCK` or `UNLOCK` the authenticating user's account profile.

Outputs

This node does not change the shared node state.

Outcomes

Single outcome path; the node updates the account status according to the configured **Lock Action**:

`LOCK`

The account is inactive and the user cannot authenticate.

`UNLOCK`

The account is active and the user can authenticate.

Errors

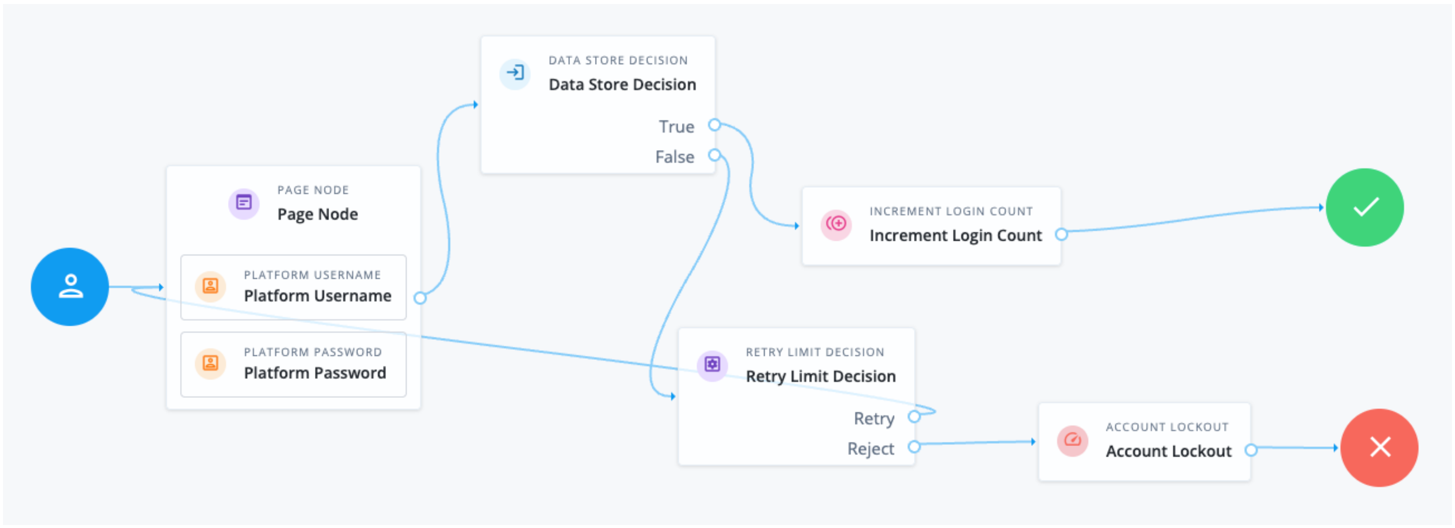
If this node fails to set the account status, it logs a `failed to set the user status inactive` warning.

This node can also throw exceptions with the following messages:

Message	Notes
`Could not get a valid username from the context`	Failed to read the `username` from the shared node state
`Could not get a valid realm from the context`	Failed to read the `realm` from the shared node state
`Could not find the identity based on the information available on context`	Failed to find the account profile with this `username` in this `realm`
`An error occurred when trying to lock out the user account`	Failed to update the account status; applies when locking and unlocking the account

Example

The following simple example uses this node with the Retry Limit Decision node to lock an account after the set number of invalid attempts:



The Retry Limit Decision node **Retry limit** (default: 3) defines the number of failed attempts before lockout.

Before using a journey like this in deployment, adapt it to reset the retry count on successful authentication.

Auth Level Decision node

Compares the current authentication level value against a configured value.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `True`
- `False`

Properties

Property	Usage
Sufficient Authentication Level	Evaluation continues along the `True` path if the current authentication level is equal to or greater than this integer; otherwise, the evaluation continues along the `False` path.

CAPTCHA node

The **CAPTCHA** node adds CAPTCHA support by verifying the response token received from the CAPTCHA provider and creating a callback for the UI to interact with.

By default, the node is configured for Google's reCAPTCHA v2.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes

Product	Available?
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

None. This node doesn't read shared state data.

Dependencies

You need to sign up for access to the reCAPTCHA API to get the API key pair required for configuring the node.

Configuration

Property	Usage
CAPTCHA Site Key *(required)*	The CAPTCHA site key supplied by the CAPTCHA provider when you sign up for access to the API.
CAPTCHA Secret Key	The CAPTCHA secret key supplied by the CAPTCHA provider when you sign up for access to the API. **Important** This property is deprecated and will be removed in a future release. Use the **CAPTCHA Secret Label Identifier** instead. If you set a **CAPTCHA Secret Label Identifier** and AM finds a matching secret in a secret store, the **CAPTCHA Secret Key** is ignored.
CAPTCHA Secret Label Identifier	An identifier used to create a *secret label* for mapping to a secret in a secret store. AM uses this identifier to create a specific secret label for this node. The secret label takes the form `am.authentication.nodes.captcha.identifier.secret` where identifier is the value of **CAPTCHA Secret Label Identifier**. The identifier can only contain alphanumeric characters `a-z`, `A-Z`, `0-9`, and periods (`.`). It can't start or end with a period. If you set a **CAPTCHA Secret Label Identifier** and AM finds a matching secret in a secret store, the **CAPTCHA Secret Key** is ignored.
CAPTCHA Verification URL	The URL used to verify the CAPTCHA submission. Possible values are: • Google: `https://www.google.com/recaptcha/api/siteverify` • hCaptcha: `https://hcaptcha.com/siteverify`

Property	Usage
CAPTCHA API URL *(required)*	The URL of the JavaScript that loads the CAPTCHA widget. Possible values are: • Google: `https://www.google.com/recaptcha/api.js` • hCaptcha: `https://hcaptcha.com/1/api.js`
Class of CAPTCHA HTML Element	The class of the HTML element required by the CAPTCHA widget. Possible values are: • Google: `g-recaptcha` • hCaptcha: `h-captcha`
ReCaptcha V3 node	If you're using Google reCAPTCHA, specify whether it's v2 or v3. Turn on for v3.
Score Threshold	If you're using Google reCAPTCHA v3 or hCaptcha, enter a score threshold. The CAPTCHA provider returns a score for each user request, based on observed interaction with your site. CAPTCHA "learns" by observing real site traffic, so scores in a staging environment or in a production deployment that has just been implemented might not be very accurate. A score of 1.0 is likely a good user interaction, while 0.0 is likely to be a bot. The threshold you set here determines whether to allow or deny access, based on the score returned by the CAPTCHA provider. Start with a threshold of 0.5. Learn more about score thresholds in the Google documentation.
Disable submission until verified	If selected, form submission is disabled until CAPTCHA verification succeeds. Default: Enabled

Outputs

None.

Outcomes

`True`

The CAPTCHA response was successfully verified.

`False`

The CAPTCHA response wasn't verified or failed verification.

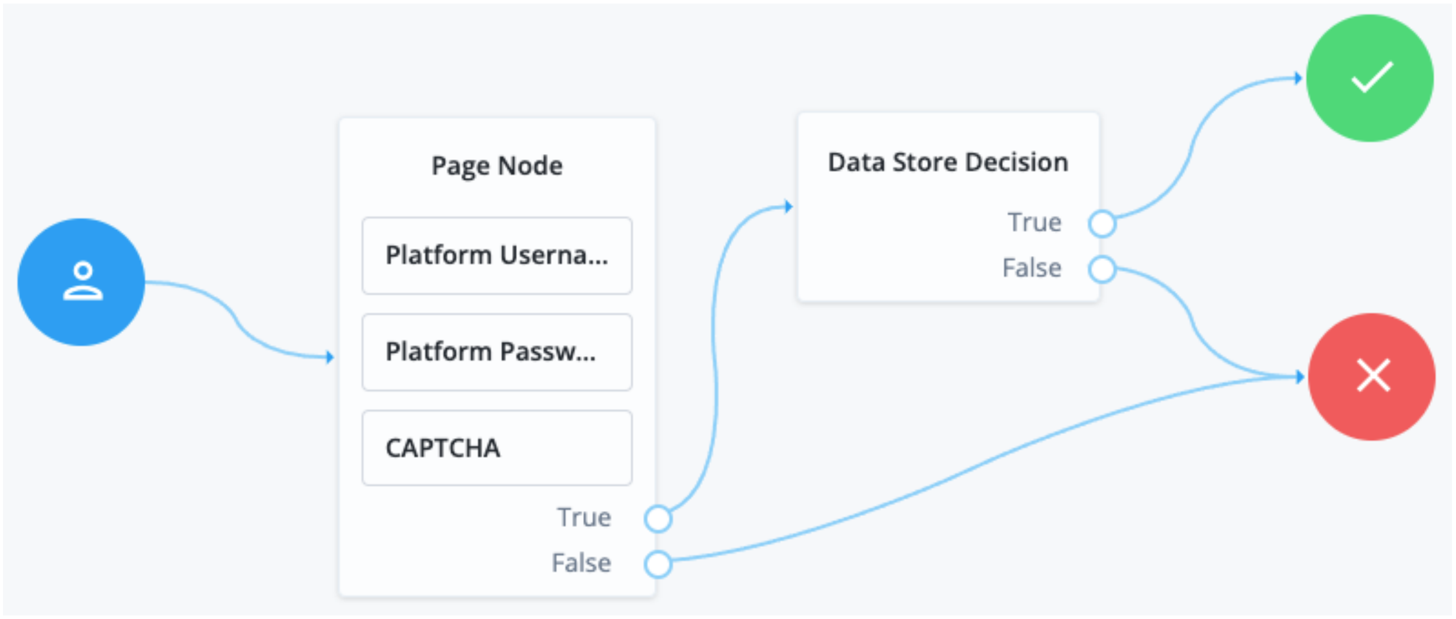
Errors

This node can throw exceptions with the following messages:

- `CAPTCHA response required for verification`
- `Unable to verify CAPTCHA response`
- `Unable to retrieve state from token response`
- `No secret key found`

Example

The following journey uses a Page node and a Data Store Decision node to collect and verify the credentials and a CAPTCHA response:



This example uses the following nodes:

- The Page node prompts the user to input their username and password:
  - The Platform Username node collects the username and stores it in the shared state.
  - The Platform Password node collects the password and stores it in the shared state.
  - The CAPTCHA node collects and verifies the CAPTCHA response.
- The Data Store Decision node uses the username and password to determine whether authentication is successful.

Legacy CAPTCHA node

Verifies the response token received from the CAPTCHA verifier, and creates a CAPTCHA callback for the UI to interact with. Default values are for Google ReCAPTCHA.

Important

This node has been superseded by the CAPTCHA node. Use that node instead.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `True` *(success)*
- `False` *(failure)*

Properties

Property	Usage
CAPTCHA Site Key *(required)*	The CAPTCHA site key supplied by the CAPTCHA provider when you sign up for access to the API.
CAPTCHA Secret Key *(required)*	The CAPTCHA secret key supplied by the CAPTCHA provider when you sign up for access to the API.
CAPTCHA Verification URL *(required)*	The URL used to verify the CAPTCHA submission. Possible values are: • Google: `https://www.google.com/recaptcha/api/siteverify` • hCaptcha: `https://hcaptcha.com/siteverify`
CAPTCHA API URL *(required)*	The URL of the JavaScript that loads the CAPTCHA widget. Possible values are: • Google: `https://www.google.com/recaptcha/api.js` • hCaptcha: `https://hcaptcha.com/1/api.js`

Property	Usage
Class of CAPTCHA HTML Element	The class of the HTML element required by the CAPTCHA widget. Possible values are: • Google: `g-recaptcha` • hCaptcha: `h-captcha`

Modify Auth Level node

Increases or decreases the current authentication level value.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

Single outcome path.

Properties

Property	Usage
Value To Add	Enter a positive integer to increase the current authentication level, or a negative integer to decrease the current authentication level by the specified value.

PingOne Protect Evaluation node

The **PingOne Protect Evaluation** node contacts PingOne to calculate the risk level and other risk-related details associated with an event.

Depending on how you configure your risk policies in PingOne, the response could return a risk score, a risk level such as high, medium, or low, and recommended actions to take, such as mitigation against bots.

For more information, refer to PingOne Protect > How it Works.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node can use shared state variables that contain the PingOne `user.id` and `user.name` as input. If these are not available, the node uses the `UserId` and `Username` variables.

This node requires that you have initialized PingOne Protect in your client application. For example, by using a PingOne Protect Initialization node node previously in the journey or by initializing the SDK within the app itself.

Dependencies

This node requires a PingOne Worker Service configuration so that it can connect to your PingOne instance and send it the necessary data to make risk evaluations.

The client application must be using Ping SDK 4.4.0 or later.

Configuration

Property	Usage
PingOne Worker Service ID	The ID of the PingOne worker service for connecting to PingOne.
Target App ID	Optional. If the user is attempting to access a PingOne application through the journey, add its v4 UUID client ID. This correlates the authentication with the application in PingOne, allowing you to filter by the **Resource Id** that matches the entered Target App ID when viewing the audit log in PingOne. For example, `12345678-abcd-4567-abcd-a123b123c123`.
Risk Policy Set ID	The ID of the risk policy in PingOne. To view risk policies in the PingOne administration console, navigate to **Threat Protection > Risk Policies**. If not specified, the environment's default risk policy set is used.

Property	Usage
Flow Type	The type of flow or event for which the risk evaluation is being carried out. Choose from: **`REGISTRATION`** Initial registration of an account. **`AUTHENTICATION`** Standard authentication for login or actions such as password change. **`ACCESS`** Verification of whether the user can access the relevant application. **`AUTHORIZATION`** Verification of whether the user is authorized to perform a specific action such as a profile change. **`TRANSACTION`** Authentication carried out in the context of a purchase or other one-time transaction. The default is `AUTHENTICATION`.
Device Sharing Type	Whether the device is shared between users or not. Choose from: • `UNSPECIFIED` • `SHARED` • `PRIVATE` The default is `SHARED`.
User Type	The type of user associated with the event. Choose from: **`PING_ONE`** User exists within the PingOne environment. **`EXTERNAL`** User exists outside PingOne, such as a federated user. The default is `EXTERNAL`.
Score Threshold	Scoring higher than this value results in evaluation continuing along the `Exceeds Score Threshold` outcome. The default is `300`.

Property	Usage
Recommended Actions	A list of recommended actions the risk evaluation could return. Each entry in the list becomes a node outcome. If the evaluation score does not exceed the **Score Threshold** value, and a recommended action is present in the response from PingOne Protect, the journey continues down the matching entry in this list. Possible values are: **`BOT_MITIGATION`** PingOne suspects the client may be automated or a bot. You should route the journey to a CAPTCHA node or similar next step to mitigate against bots. **`AITM_MITIGATION`** PingOne suspects an adversary-in-the-middle (AitM) attack. You should route the journey to the failure node, and consider locking the account, and force a password change to mitigate against these attacks.
Pause Behavioral Data	After receiving the device signal, instruct the client to pause collecting behavioral data. Default: Selected
Node State Attribute For User ID	The node state variable that contains the `user.id` as it appears in PingOne. If left blank, the node uses the current context `UserId` as the `user.id`.
Node State Attribute For Username	The node state variable that contains the `user.name` as it appears in PingOne. If left blank, the node uses the current context `Username` as the `user.name`.
Store Risk Evaluation	Stores the risk evaluation response in the transient node state under a key named `PingOneProtectEvaluationNode.RISK`. The default is not enabled. **Note** The key is empty if the node is unable to retrieve a risk evaluation from PingOne.

Outputs

If you enable the **Store Risk Evaluation** property, the node outputs the risk evaluation response JSON in a state variable named `PingOneProtectEvaluationNode.RISK`.

Outcomes

`High`

The risk evaluation level is considered high.

`Medium`

The risk evaluation level is considered medium.

`Low`

The risk evaluation level is considered low.

`Exceeds Score Threshold`

The score returned is higher than the configured threshold.

`Failure`

The risk evaluation could not be completed.

Recommended Actions

The risk evaluation recommended a mitigation action to take, and it matched a value in the **Recommended Actions** list.

Currently, the only value possible is `BOT_MITIGATION`, which recommends you check for the presence of a human, for example, by using a CAPTCHA node.

`ClientError`

The client returned an error when attempting to capture the data to perform a risk evaluation.

Outcome precedence

Evaluation of the journey continues along an outcome based on the response received and which fields are present in it, as follows:

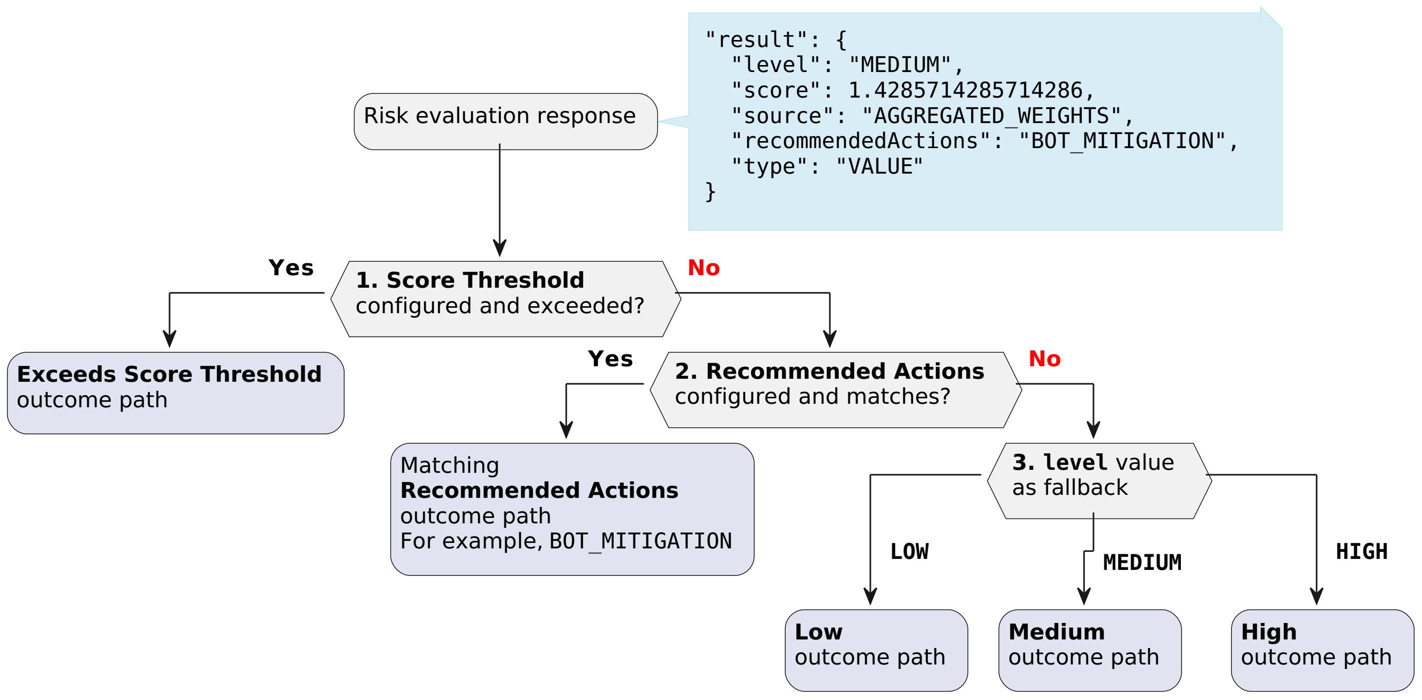

Risk evaluation response
"result": {
  "level": "MEDIUM",
  "score": 1.4285714285714286,
  "source": "AGGREGATED_WEIGHTS",
  "recommendedActions": "BOT_MITIGATION",
  "type": "VALUE"
}
1. Score Threshold configured and exceeded?
Yes
No
Exceeds Score Threshold outcome path
2. Recommended Actions configured and matches?
Yes
No
Matching Recommended Actions outcome path For example, BOT_MITIGATION
3. level value as fallback
LOW
MEDIUM
HIGH
Low outcome path
Medium outcome path
High outcome path

Figure 1. Risk evaluation outcome path precedence

1. If you have configured the **Score Threshold** property and the result contains a score that exceeds it, evaluation continues along the `Exceeds Score Threshold` outcome path.

2. If you have *not* configured the **Score Threshold** property, or the score does not exceed it, but *have* added a value in the **Recommended Actions** list that matches one in the response, evaluation continues along the relevant dynamic outcome path. For example, the `BOT_MITIGATION` outcome path.

3. If you have *not* configured the **Score Threshold** property, or the score does not exceed it, and have *not* added a matching value in the **Recommended Actions** list, then evaluation continues along the relevant `level` path, one of `Low`, `Medium`, or `High`.

Example

The following example journey leverages PingOne Protect functionality to perform a risk evaluation on a client app. The client app is built using the ForgeRock SDKs.

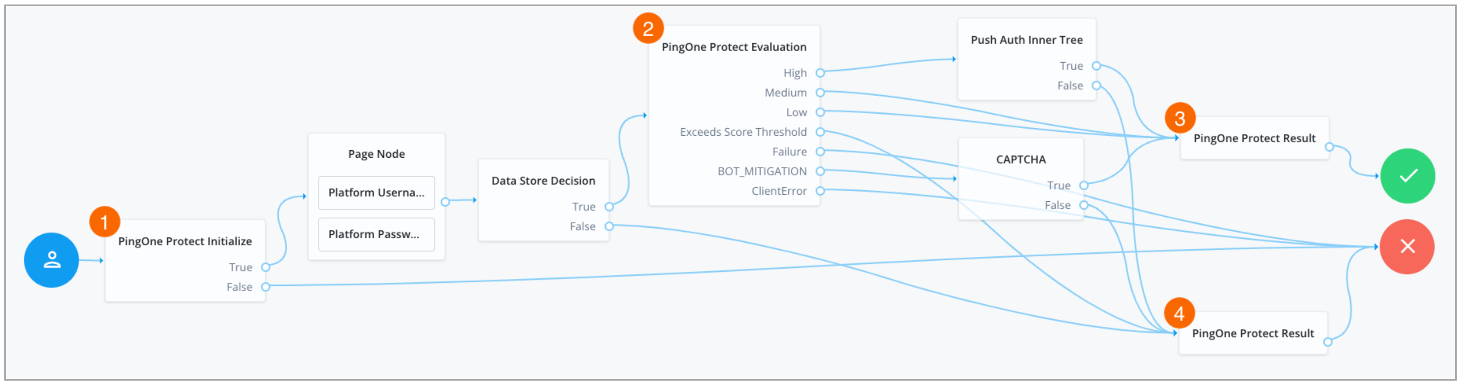


Figure 2. Example PingOne Protect journey

- 1 The PingOne Protect Initialization node instructs the SDK to initialize the PingOne Protect Signals API with the configured properties.

> **Tip**
>
> Initialize the PingOne Protect Signals API as early in the journey as possible, before any user interaction. + This enables it to gather sufficient contextual data to make an informed risk evaluation.

- The user enters their credentials, which are verified against the identity store.
- 2 The PingOne Protect Evaluation node performs a risk evaluation against a risk policy in PingOne.

The example journey continues depending on the outcome:

`High`

The journey requests that the user respond to a push notification.

`Medium` ***or*** **`Low`**

The risk is not significant, so no further authentication factors are required.

`Exceeds Score Threshold`

The score returned is higher than the configured threshold and is considered too risky to complete successfully.

`Failure`

The risk evaluation could not be completed, so the authentication attempt continues to the **Failure** node.

`BOT_MITIGATION`

The risk evaluation returned a recommended action to check for the presence of a human, so the journey continues to a CAPTCHA node.

`ClientError`

The client returned an error when attempting to capture the data to perform a risk evaluation, so the authentication attempt continues to the **Failure** node.

- 3 An instance of the PingOne Protect Result node returns the `Success` result to PingOne, which can be viewed in the console to help with analysis and risk policy tuning.
- 4 A second instance of the PingOne Protect Result node returns the `Failed` result to PingOne, which can be viewed in the console to help with analysis and risk policy tuning.

PingOne Protect Initialization node

The **PingOne Protect Initialization** node instructs the SDK to initialize the embedded PingOne Protect SDK on the client device using the configuration provided by the node properties.

For more information, refer to Threat Protection using PingOne Protect.

You can only initialize the PingOne Protect SDK on the client device once. Attempting to initialize the SDK with a different configuration will not override the initial settings.

Tip

You should initialize the PingOne Protect SDK on the client device as early as possible so that it can gather sufficient contextual information to make risk evaluations.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Important

This node is not currently compatible with the following user interfaces:

- The **XUI** interface provided by standalone AM deployments.
- The **Platform UI** interface provided by Advanced Identity Cloud and ForgeOps deployments.

You can only use this node in client applications built using the ForgeRock SDK. Refer to Integrate with PingOne Protect for risk evaluations.

Inputs

This node has no required predecessor nodes.

It does not read from the shared node state.

Dependencies

This node requires a **PingOne Worker Service** configuration so that it can connect to your PingOne instance and send it the necessary data to make risk evaluations as part of the journey.

Find information on the properties used by the service in PingOne Worker service.

The client application must be using ForgeRock SDK 4.4.0 or later.

Configuration

Property	Usage
PingOne Worker Service ID	The ID of the PingOne worker service for connecting to PingOne.
Enable SDK Logs	When enabled, output SDK log messages in the developer console. Default: Disabled

Property	Usage
Device Attributes To Ignore	A list of device attributes you want to exclude from the results when collecting device signals. These attributes will not be sent to PingOne to perform evaluations, which might limit its ability to create accurate results. Some examples of attributes the client might obtain from the device include: • `BATTERY_LEVEL` • `CPU_ARCHITECTURE` • `DEVICE_MODEL` • `DEVICE_VENDOR` • `GPS_SUPPORTED` • `HAS_CHROME_APP` • `IS_ACCEPT_COOKIES` • `NAVIGATOR_USER_AGENT` • `OS_NAME` • `OS_VERSION` • `RESOLUTION` • `TOUCH_SUPPORT` **Note** The attributes collected vary depending on the OS of the client. For example, an Android device might provide different attributes to a JavaScript app running on Windows.
Custom Host	Deprecated. We recommend that you do not change this property.
Lazy Metadata	When enabled, calculate metadata on demand. When not enabled, metadata is calculated automatically after initialization. Default: Disabled
Collect Behavioral Data	When enabled, collect behavioral data. When not enabled, behavioral data is not collected. Default: Enabled
Disable Hub	When selected, the client stores device data in the browser’s `localStorage` only. When not selected, an iframe is used. Default: Not selected
Device Key Rsync Intervals (days)	Number of days that device attestation can rely upon the device fallback key. Default: `14`
Enable Trust	Tie the device payload to a non-extractable crypto key stored in the browser for content authenticity verification
Disable Tags	When selected, the client does not collect tag data. Tags are used to record the pages the user visited, forming a browsing history. Default: Not selected

Outputs

The node sends a `PingOneProtectInitializeCallback` to the client application.

The ForgeRock SDKs consume this callback and initialize the PingOne Protect functionality so it can start gathering the data it needs to make risk evaluations.

Outcomes

`True`

The client application confirmed successful receipt of the configuration.

`False`

The client application *did not* confirm successful receipt of the configuration or returned a client error.

Example

The following example journey leverages PingOne Protect functionality to perform a risk evaluation on a client app. The client app is built using the ForgeRock SDKs.

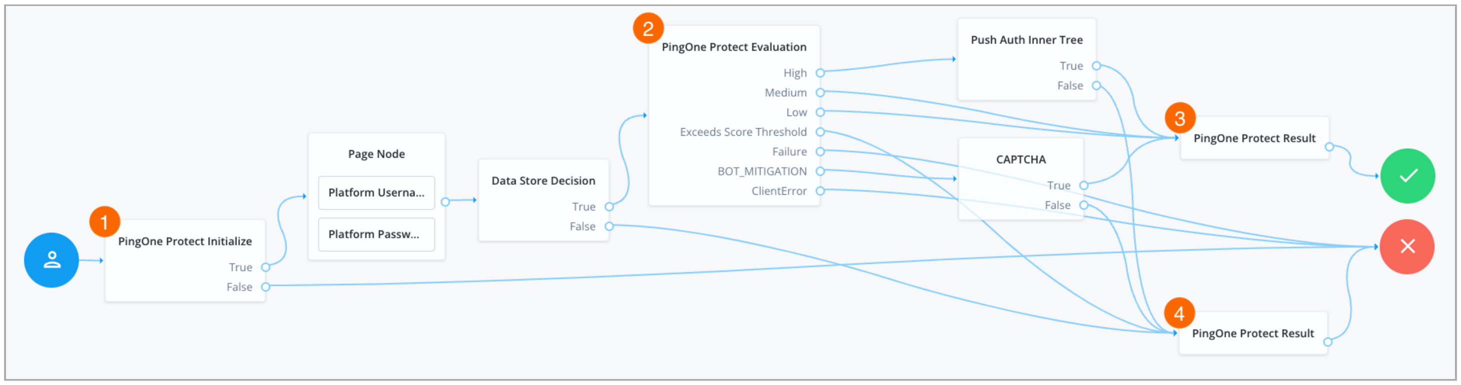


Figure 1. Example PingOne Protect journey

- 1 The PingOne Protect Initialization node instructs the SDK to initialize the PingOne Protect Signals API with the configured properties.

  Tip

  Initialize the PingOne Protect Signals API as early in the journey as possible, before any user interaction.
  + This enables it to gather sufficient contextual data to make an informed risk evaluation.

- The user enters their credentials, which are verified against the identity store.
- 2 The PingOne Protect Evaluation node performs a risk evaluation against a risk policy in PingOne.

  The example journey continues depending on the outcome:

  `High`

  The journey requests that the user respond to a push notification.

`Medium` *or* `Low`

The risk is not significant, so no further authentication factors are required.

`Exceeds Score Threshold`

The score returned is higher than the configured threshold and is considered too risky to complete successfully.

`Failure`

The risk evaluation could not be completed, so the authentication attempt continues to the **Failure** node.

`BOT_MITIGATION`

The risk evaluation returned a recommended action to check for the presence of a human, so the journey continues to a CAPTCHA node.

`ClientError`

The client returned an error when attempting to capture the data to perform a risk evaluation, so the authentication attempt continues to the **Failure** node.

- 3 An instance of the PingOne Protect Result node returns the `Success` result to PingOne, which can be viewed in the console to help with analysis and risk policy tuning.
- 4 A second instance of the PingOne Protect Result node returns the `Failed` result to PingOne, which can be viewed in the console to help with analysis and risk policy tuning.

PingOne Protect Result node

The **PingOne Protect Result** node updates the risk evaluation configuration, or modify the completion status of the resource while the risk evaluation is still in progress.

You can check the results of the evaluation in the PingOne admin console, by filtering for `Risk Evaluation Updated` event types.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Important

This node is not currently compatible with the following user interfaces:

- The **XUI** interface provided by standalone AM deployments.
- The **Platform UI** interface provided by Advanced Identity Cloud and ForgeOps deployments.

You can only use this node in client applications built using the ForgeRock SDK. Refer to Integrate with PingOne Protect for risk evaluations.

Inputs

This node requires that you have initialized PingOne Protect in your client application. For example, by using a PingOne Protect Evaluation node previously in the journey or by initializing the SDK within the app itself.

Dependencies

This node requires a **PingOne Worker Service** configuration so that it can connect to your PingOne instance and send it the necessary data to make risk evaluations as part of the journey.

Configuration

Property	Usage
Completion Status	Report the status of the journey back to PingOne. Choose from: • `FAILED` • `SUCCESS`

Outputs

This node does not change the shared node state.

Outcomes

Single outcome path.

The node attempts to update the PingOne server but continues along the single outcome without confirming the server received the update.

Example

The following example journey leverages PingOne Protect functionality to perform a risk evaluation on a client app. The client app is built using the ForgeRock SDKs.

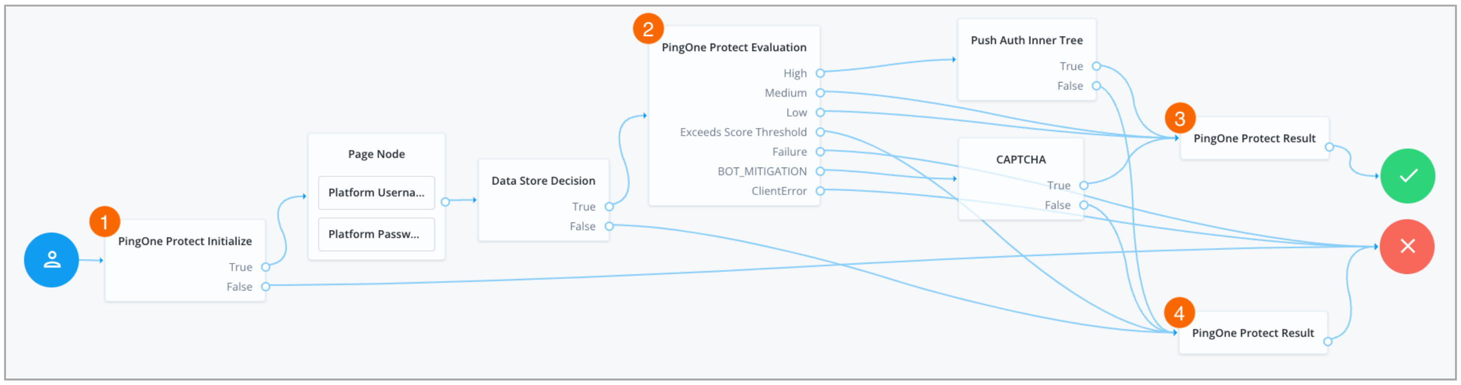


Figure 1. Example PingOne Protect journey

- 1 The PingOne Protect Initialization node instructs the SDK to initialize the PingOne Protect Signals API with the configured properties.

> **Tip**
>
> Initialize the PingOne Protect Signals API as early in the journey as possible, before any user interaction. + This enables it to gather sufficient contextual data to make an informed risk evaluation.

- The user enters their credentials, which are verified against the identity store.
- 2 The PingOne Protect Evaluation node performs a risk evaluation against a risk policy in PingOne.

The example journey continues depending on the outcome:

`High`

The journey requests that the user respond to a push notification.

`Medium` *or* `Low`

The risk is not significant, so no further authentication factors are required.

`Exceeds Score Threshold`

The score returned is higher than the configured threshold and is considered too risky to complete successfully.

`Failure`

The risk evaluation could not be completed, so the authentication attempt continues to the **Failure** node.

`BOT_MITIGATION`

The risk evaluation returned a recommended action to check for the presence of a human, so the journey continues to a CAPTCHA node.

`ClientError`

The client returned an error when attempting to capture the data to perform a risk evaluation, so the authentication attempt continues to the **Failure** node.

- 3 An instance of the PingOne Protect Result node returns the `Success` result to PingOne, which can be viewed in the console to help with analysis and risk policy tuning.
- 4 A second instance of the PingOne Protect Result node returns the `Failed` result to PingOne, which can be viewed in the console to help with analysis and risk policy tuning.

Behavioral nodes

Increment Login Count node

The **Increment Login Count** node increments the successful login count property of a managed object.

Use the Login Count Decision node to change the flow of the journey based on the count.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node's **Identity Attribute** specifies the property it requires in the incoming node state. It uses this property to access the managed object.

Dependencies

This node depends on IDM to store the managed object.

Configuration

Property	Usage
Identity Attribute	The attribute used to identify the managed object in IDM. Default: `userName`

Outputs

This node does not change the shared node state.

Outcomes

Single outcome path; on success, this node increments the managed object's `loginCount`.

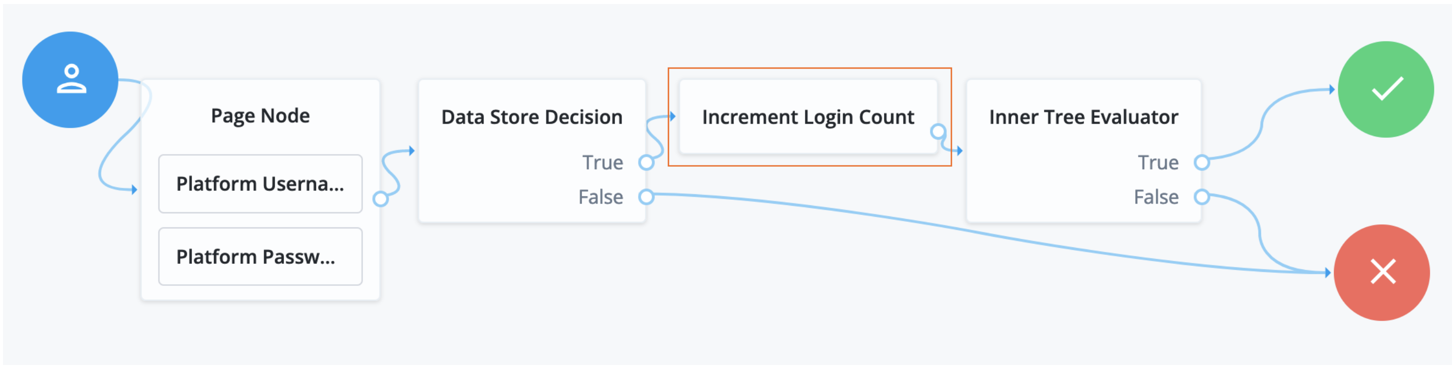
Errors

If this node fails to access the managed object, it throws an exception with a `No object to increment` message.

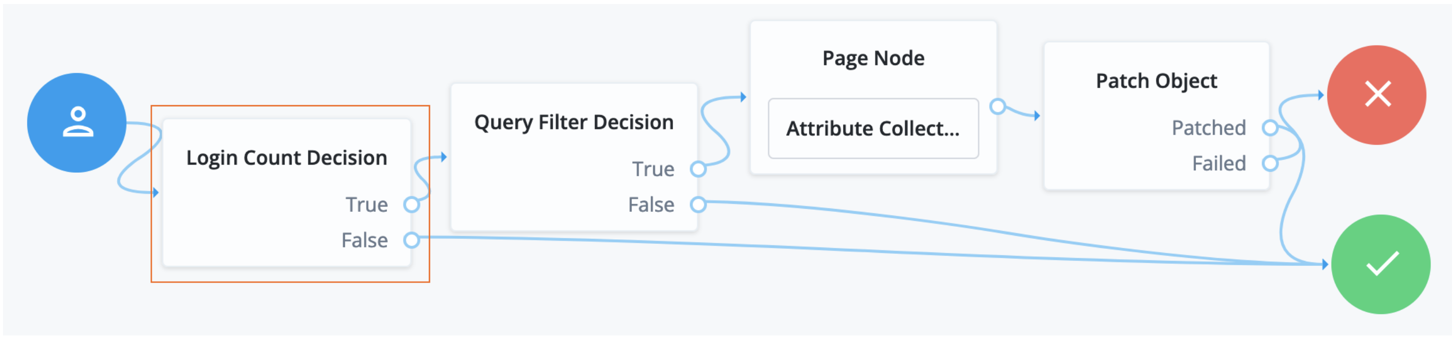
If this node fails to increment the login count, it logs an `Unable to increment login count` warning message.

Example

The following journey uses the Increment Login Count node to update the login count on successful authentication:



- The Platform Username node injects the `userName` into the shared node state.
- The Data Store Decision node determines whether authentication is successful.
- The Increment Login Count node (outlined in the image) updates the login count.
- The Inner Tree Evaluator node invokes the following nested journey for progressive profiling:



- The Login Count Decision node triggers the rest of the journey depending on the login count and its settings.
- The Query Filter Decision node determines whether managed object profile fields are still missing.

- The Page node requests additional input for the profile.
- The Patch Object node stores the additional input in the managed object profile.

Login Count Decision node

The **Login Count Decision** node triggers an action when a user's successful login count property reaches a specified number.

Use the Increment Login Count node to set the login count on successful authentication.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node's **Identity Attribute** specifies the property it requires in the incoming node state. It uses this property to access the managed object.

Dependencies

This node depends on IDM to store the managed object.

Configuration

Property	Usage
Interval	Trigger the `True` outcome depending on this setting, the **Amount**, and the login count: **AT** Proceed to `True` when the login count matches the **Amount** setting. **EVERY** Proceed to `True` every time the login count reaches a multiple of the **Amount** setting. Default: `AT`
Amount	The login count to trigger a `True` outcome depending on the **Interval**. Default: 25
Identity Attribute	The attribute used to identify the managed object in IDM. Default: `userName`

Outputs

This node does not change the shared node state.

Outcomes

`True`

The login count reached **Amount**, and the **Interval** setting triggered this outome.

`False`

All other cases.

Errors

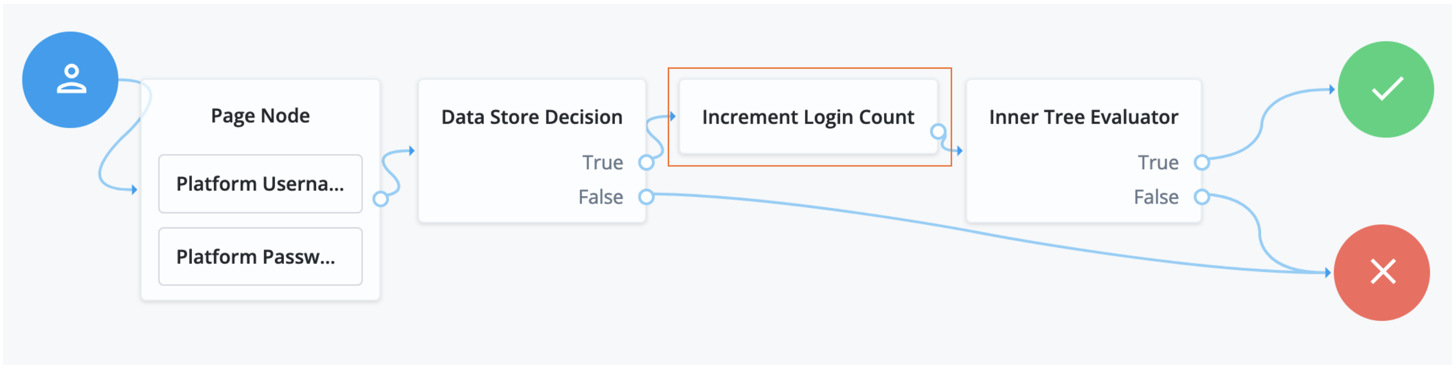
This node can throw exceptions with the following messages:

Message	Notes
`<Identity Attribute> not present in state`	Failed to read the specified **Identity Attribute** in the shared node state
`Failed to retrieve existing object`	Failed to find the managed object using the **Identity Attribute** value from the shared node state

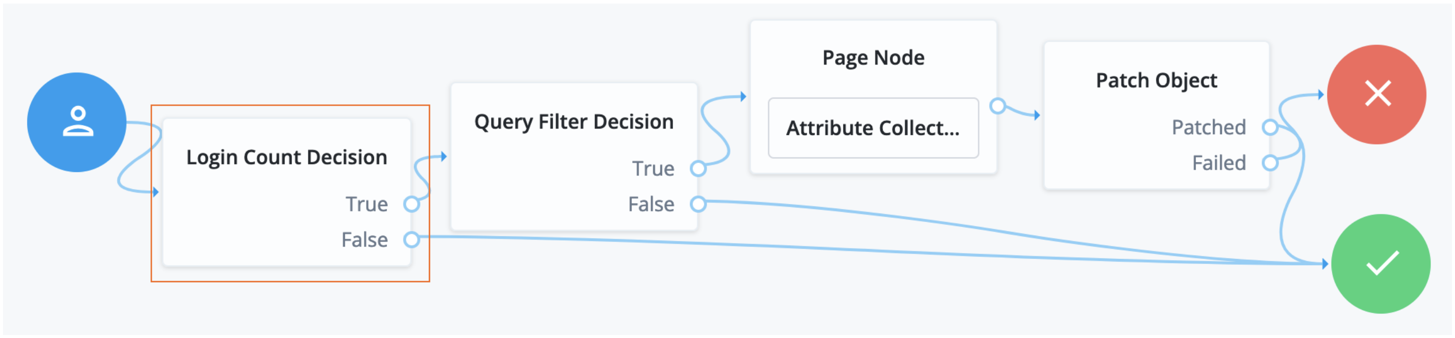
Message	Notes
`Retrieve login not found`	Failed to read the managed object's login count

Example

The following journey uses the Increment Login Count node to update the login count on successful authentication:



- The Platform Username node injects the `userName` into the shared node state.
- The Data Store Decision node determines whether authentication is successful.
- The Increment Login Count node (outlined in the image) updates the login count.
- The Inner Tree Evaluator node invokes the following nested journey for progressive profiling:



- The Login Count Decision node triggers the rest of the journey depending on the login count and its settings.
- The Query Filter Decision node determines whether managed object profile fields are still missing.
- The Page node requests additional input for the profile.
- The Patch Object node stores the additional input in the managed object profile.

Contextual nodes

Certificate Collector node

Collects an X.509 digital certificate from the request to use the certificate as authentication credentials.

To validate the certificate, use a Certificate Validation node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `Collected`
- `Not Collected`

Evaluation continues through the `Collected` path if certificate collection is successful; otherwise, evaluation continues on the `Not Collected` path.

Properties

Property	Usage
Certificate Collection Method	Specifies how to collect the certificate from the request. Possible values are: **`Request`** Look for the certificate in the request. Use this value if TLS termination happens at the container where AM runs. **`Header`** Looks for the certificate in the HTTP header name specified in the **HTTP Header Name for the Client Certificate** property. Use this value if TLS termination happens in a proxy or load balancer outside the container where AM runs. **`Either`** Looks for the certificate in the request; if AM cannot find it in the request, AM looks for the certificate in the HTTP header specified in the **HTTP Header Name for the Client Certificate** property. Default: `Either`
HTTP Header Name for the Client Certificate	Specifies the name of the HTTP header containing the certificate when the **Certificate Collection Method** property is configured to `Header` or `Either`. Default: No value specified.
Trusted Remote Hosts	Specifies a list of IP addresses trusted to supply certificates on behalf of the authenticating client, such as load balancers doing TLS termination. If no value is specified, AM rejects certificates supplied by remote hosts. If you specify the `any` value, AM trusts certificates on behalf of the authenticating client supplied by any remote host. Default: No value specified.

Certificate User Extractor node

Extracts a value from the certificate collected by the Certificate Collector node, and searches for it in the identity store. The goal is to match the certificate with a user in the identity store.

The extracted value is stored in the `username` key in the shared node state.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes

Product	Available?
Ping Identity Platform (self-managed)	Yes

Outcomes

- `Extracted`
- `Not Extracted`

Evaluation continues through the `Extracted` path if AM finds a match for the certificate in the identity store; otherwise, evaluation continues on the `Not Extracted` path.

Properties

Property	Usage
Certificate Field Used to Access User Profile	Specifies the field in the certificate that AM uses to search for the user in the identity store. Possible values are: • `Subject DN` • `Subject CN` • `Subject UID` • `Email Address` • `Other` • `None` If you select `Other`, provide an attribute name in the **Other Certificate Field Used to Access User Profile** property. Select `None` if you want to specify an alternate way of looking up the user profile in the **SubjectAltNameExt Value Type to Access User Profile** property. Default: `Subject CN`
Other Certificate Field Used to Access User Profile	Specifies a custom certificate field to use as the base of the user search.
SubjectAltNameExt Value Type to Access User Profile	Specifies how to look up the user profile: **None** AM uses the value specified in the **Certificate Field Used to Access User Profile** or the **Other Certificate Field Used to Access User Profile** properties when looking up the user profile. **RFC822Name** AM looks up the user profile using the value of the `RFC822Name` field. **UPN** AM looks up the user profile as the User Principal Name attribute used in Active Directory. Default: `None`

Certificate Validation node

The **Certificate Validation** node validates a digital X.509 certificate collected by the Certificate Collector node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node requires an `X509Certificate` property in the incoming node state.

Implement the Certificate Collector node as input to the Certificate Validation node.

Configuration

Property	Usage
Match Certificate in LDAP	When enabled, AM matches the certificate collected with the one stored in an LDAP directory entry. You define the name of this entry and additional security-related properties later in the node configuration. Default: Disabled
Check Certificate Expiration	When enabled, AM checks whether the provided certificate has expired. Default: Disabled
Subject DN Attribute Used to Search LDAP for Certificates	The attribute that AM uses to search the LDAP directory for the certificate. The search filter is based on this attribute and the value of the Subject DN as it appears in the certificate. Default: `CN`
Match Certificate to CRL	When enabled, AM checks whether the certificate has been revoked according to a Certificate Revocation List (CRL) in the LDAP directory. Define related CRL properties later in the node configuration. Default: Disabled.

Property	Usage
Issuer DN Attribute(s) Used to Search LDAP for CRLs	The name of the attribute or attributes in the issuer certificate that AM uses to locate the CRL in the LDAP directory. • If you specify only one attribute here, the LDAP search filter used is `(attr-name=attr-value-in-subject-DN)`. For example, if the subject DN of the issuer certificate is `C=US, CN=Some CA, serialNumber=123456`, and the attribute specified is `CN`, then AM uses a search filter of `(CN=Some CA)` to locate the CRL. • Specify several CRLs for the same CA issuer in a comma-separated list (`,`) where the names are in the same order in which they appear in the subject DN. In this case, the LDAP search filter used is `(attr1=attr1-value-in-subject-DN,attr2=attr2-value-in-subject-DN,…)`, and so on. For example, if the subject DN of the issuer certificate is `C=US, CN=Some CA, serialNumber=123456`, and the attributes specified are `CN,serialNumber`, then the LDAP search filter used to find the CRL is `(CN=Some CA,serialNumber=123456)`. Default: `CN`
HTTP Parameters for CRL Update	Parameters that AM includes in any HTTP CRL call to the CA that issued the certificate. If the client or CA certificate includes the `IssuingDistributionPoint` extension, AM uses this information to retrieve the CRL from the distribution point. Add the parameters as key-value pairs in a comma-separated list (`,`). For example, `param1=value1,param2=value2`.
Cache CRLs in Memory	When enabled, AM caches CRLs in memory. If this option is enabled, **Update CA CRLs from CRLDistributionPoint** must also be enabled. Default: Enabled
Update CA CRLs from CRLDistributionPoint	When enabled, AM fetches new CA CRLs from the CRL Distribution Point and updates them in the LDAP directory. If the CA certificate includes either the `IssuingDistributionPoint` or the `CRLDistributionPoint` extensions, AM attempts to update the CRLs when they're out of date. Default: Enabled
OCSP Validation	When enabled, AM checks the validity of certificates using the Online Certificate Status Protocol (OCSP). The AM instance must have internet access, and you must configure OSCP for AM under **Configure** > **Server Defaults** > **Security** > **Online Certificate Status Protocol Check**. Default: Disabled

Property	Usage
LDAP Server Where Certificates are Stored	The LDAP server that holds certificates. Enter the server details in the format `ldap-server:port` . To associate multiple AM servers in a site with corresponding LDAP servers, use the format `am_server|ldapserver:_port_` . For example, `am.example.com|ldap1.example.com:636` .
LDAP Search Start or Base DN	Valid base DN for the LDAP search, such as `dc=example,dc=com` . To associate AM servers with different search base DNs, use the format `am_server|base_dn` . For example, `am.example.com|dc=example,dc=com` `openam1.test.com|dc=test,dc=com` .
LDAP Server Authentication User and **LDAP Server Authentication Password**	The credentials used to connect to the LDAP directory that holds the certificates. If you enable mTLS, the node ignores these credentials. Default Authentication User: `cn=Directory Manager`
mTLS Enabled	Enables mTLS (mutual TLS) between AM and the directory server. When mTLS is enabled, the node ignores the values for **LDAP Server Authentication User** and **LDAP Server Authentication Password**. If you enable this property, you must: • Enable **Use SSL/TLS for LDAP Access**. • Provide an **mTLS Secret Label Identifier**. Default: Disabled
mTLS Secret Label Identifier	An identifier used to create a secret label for mapping to the mTLS certificate in the secret store. AM uses this identifier to create a specific secret label for this node. The secret label takes the form `am.authentication.nodes.certificate.validation.mtls.identifier.cert` , where `identifier` is the value of **mTLS Secret Label Identifier**. The label can only contain alphanumeric characters (`a-z` , `A-Z` , `0-9`) and periods (`.`). It can't start or end with a period. For greater security, you should rotate certificates periodically. When you rotate a certificate, update the corresponding mapping in the realm secret store configuration to reflect this identifier. When you rotate a certificate, AM closes any existing connections using the old certificate. A new connection is selected from the connection pool and no server restart is required.
Use SSL/TLS for LDAP Access	When enabled, AM uses SSL/TLS to access the LDAP directory. Make sure that AM trusts the certificate from the LDAP server when enabling this option. Default: Disabled

Outputs

This node doesn't put anything into the shared state.

Outcomes

`True`

The node could validate the certificate.

When the outcome is `True`, add a Certificate User Extractor node to extract the values of the certificate.

`False`

The node couldn't validate the certificate. The journey follows this path when the node can't validate the certificate and no more specific outcome is available.

`Not found`

The **Match Certificate in LDAP** property is enabled, but the certificate wasn't found in the LDAP store.

`Expired`

The **Check Certificate Expiration** property is enabled, and the certificate has expired.

`Path Validation Failed`

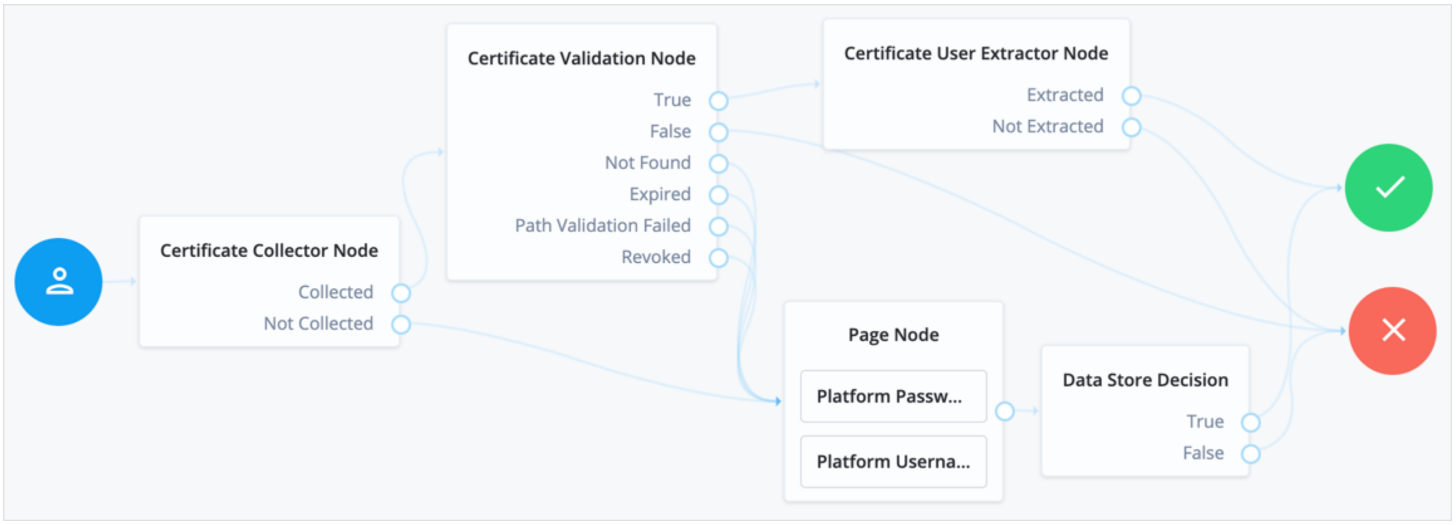
The **Match Certificate to CRL** property is enabled, and the certificate path is invalid.

`Revoked`

The **OCSP Validation** property is enabled, and the certificate has been revoked.

Example

The following is an example of how to use the certificate nodes in a Ping Identity Platform authentication journey. Note that all the failure outcomes of the Certificate Validation node are linked so that the user provides a username and password, but you could choose different authentication methods for each outcome:



Cookie Presence Decision node

Checks that a named cookie is present in the incoming authentication request.

This node does not check the value of the named cookie, only that it exists.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `True`
- `False`

Properties

Property	Usage
Name of Cookie *(required)*	Evaluation continues along the `True` path if the named cookie is present in the incoming authentication request; otherwise, evaluation continues along the `False` path.

Device Geofencing node

Compares any collected device location metadata with the trusted locations configured in the authentication node.

Use this node with the Device Profile Collector node to determine if the authenticating user's device is located within range of configured, trusted locations.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes

Product	Available?
Ping Identity Platform (self-managed)	Yes

Outcomes

- `Inside`
- `Outside`

Evaluation continues along the `Inside` path if the collected location is within the specified range of a configured trusted location; otherwise, evaluation continues along the `Outside` path.

Properties

Property	Usage
Trusted Locations *(required)*	Specify the latitude and longitude of at least one trusted location. Separate the values with a comma; for example, `37.7910855,-122.3951663`.
Geofence Radius (km)	Specifies the maximum distance, in kilometers, that a device can be from a configured trusted location. The distance is calculated point-to-point.

Device Location Match node

Compares any collected device location metadata with that stored in the user's profile.

Use this node with the Device Profile Collector node to determine if the authenticating user's device is located within range of somewhere they have authenticated from, and saved, previously.

You must establish the identity of the user before attempting to match locations.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `True`
- `False`
- `Unknown Device`

Evaluation continues along the `True` path if the collected location is within the specified range of saved location data; otherwise, evaluation continues along the `False` path.

If the user has no saved device profiles or the identity of the user has not been established, evaluation continues along the `Unknown Device` path.

Properties

Property	Usage
Maximum Radius (km)	Specifies the maximum distance, in kilometers, that a device can be from a previously saved location. The distance is calculated point-to-point.

Device Match node

The **Device Match** node compares collected device metadata with that stored in the user's profile.

Use this node with the Device Profile Collector node to check whether the user is authenticating with a previously saved, trusted device.

The Device Match node supports the following methods of comparison:

- Built-in matching

  The node handles the comparison and matching. You configure the acceptable variance and the maximum age for device profiles.

- Custom matching

  Create scripts to compare captured device data against trusted device profiles.

  AM includes a customizable template script. In the AM admin UI, go to **Realms** > ***Realm Name*** > **Scripts**, and click **Device Profile Match Template - Decision Node Script**.

  > **Tip**
  >
  > For a comprehensive sample script with instructions for its use and a development toolkit, go to the GitHub sample repository.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

The node reads the `username` from the shared state. Implement the following node before this node in the journey:

- Username Collector node (standalone AM)
- Platform Username node (Ping Identity Platform deployment)

This node also reads collected device metadata from the shared state. Implement a Device Profile Collector node earlier in the journey to collect metadata for the current device.

If **Use Custom Matching Script** is enabled, the inputs depend on the script.

Dependencies

If **Use Custom Matching Script** is enabled, the dependencies depend on the script.

Configuration

Property	Usage
Acceptable Variance	The maximum number of acceptable device attribute differences for a match. Default: `0` (all attributes must match)
Expiration	The maximum age in days a saved profile is valid for comparison. The node ignores older device profiles saved to the user's account when comparing device profiles with the collected metadata. Default: `30` (days)
Use Custom Matching Script	Enable this option to use a custom script instead of built-in matching to compare the collected metadata with saved device profiles. When enabled, the node ignores the **Acceptable Variance** and **Expiration** settings. The script type must be `Decision node script for authentication trees` (standalone AM) or `Journey Decision Node` (Ping Identity Platform deployment). Default: false

Property	Usage
Custom Matching Script	Select the custom script to use when **Use Custom Matching Script** is enabled. Only scripts of type `Decision node script for authentication trees` (standalone AM) or `Journey Decision Node` (Ping Identity Platform deployment) appear in the list. Default: `Authentication Tree Decision Node Script`

Outputs

This node does not change the shared state on its own.

If the node uses a **Custom Matching Script**, the output is determined by the script.

Outcomes

`True`

The collected device metadata matches a saved profile within the configured variance.

`False`

The collected device metadata doesn't match a saved profile, or another error occurred.

`Unknown Device`

The journey follows this outcome path in the following situations:

- The user has no saved trusted device profiles.
- The user identity hasn't yet been established.
- The acceptable device variance matches, but the device ID no longer matches.

  The device ID is randomly generated and stored in the local browser cache. If the cache is cleared, the device ID can change.

Errors

This node logs the following warning messages:

`script outcome error`

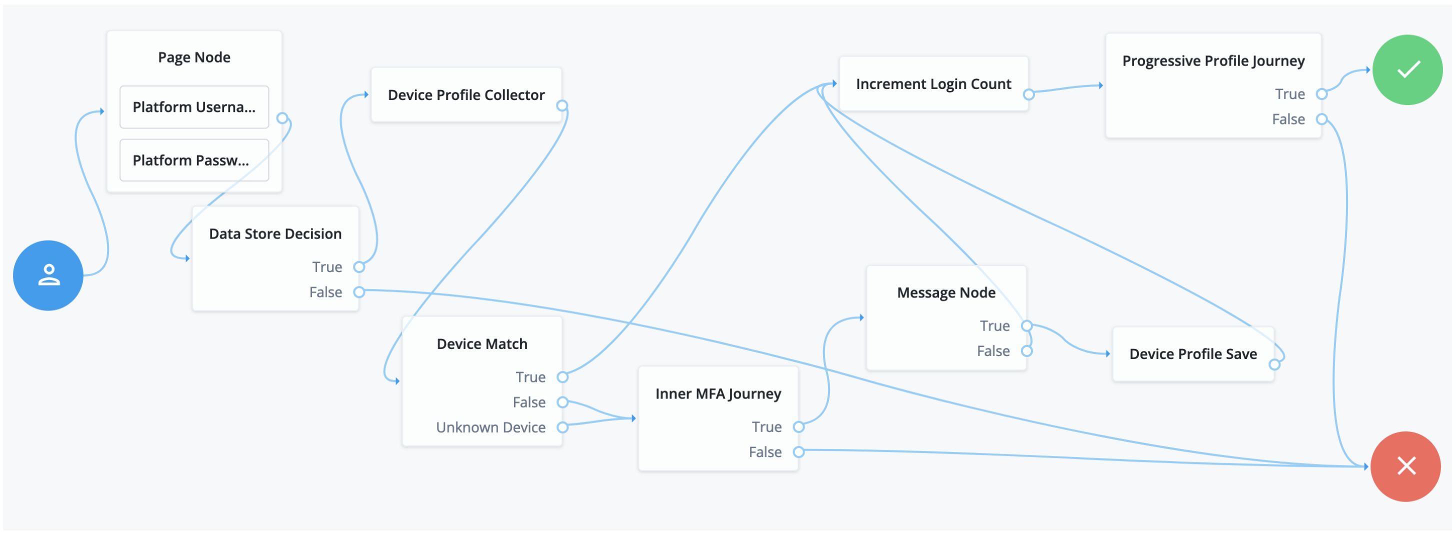
The script failed to set the `outcome` field to a string.

`error evaluating the script`

The script failed to complete. Refer to the logs for details.

Example

The following journey authenticates the user and checks whether the current device is trusted. If the device isn't trusted yet, the journey requires an additional authentication factor and lets the user opt to trust the device:



- The Page node with the Platform Username node and Platform Password node prompt the user for their credentials.
- The Data Store Decision node confirms the user's credentials.
- The Device Profile Collector node collects metadata about the current device.
- The **Device Match** node compares saved device profiles with the current device.
- The **Inner MFA Journey**, an Inner Tree Evaluator node, requires an additional authentication factor.
- The Message node prompts the user with an option to trust the current device.
- The Device Profile Save node saves the current device profile.
- The Increment Login Count node updates the number of successful authentications.
- The **Progressive Profile Journey**, an Inner Tree Evaluator node, invokes a journey to collect additional profile data.

Device Profile Collector node

Gathers metadata about the device used to authenticate.

The node sends a `DeviceProfileCallback` callback. Find more information, in Interactive callbacks.

When used with the ForgeRock SDKs, the node can collect the following:

Device Metadata

Information such as the platform, versions, device name, hardware information, and the brand of the device being used.

The captured data is in JSON format, and stored in the authentication shared state in a variable named `forgeRock.device.profile`.

Device Location

Provides the last known latitude and longitude of the device's location.

The captured data is in JSON format, and stored in the authentication shared state in a variable named `forgeRock.device.location`.

The collection of geographical information requires end-user approval. A browser function drives this process. A pop-up displays, prompting for access to share the geographical location. The browser connection must be secure.

Important

It is up to you what information you collect from users and devices.
Always use data responsibly and provide your users with appropriate control over data they share with you.
You are responsible for complying with any regulations or data protection laws.

In addition to the collected metadata, an `identifier` string in the JSON uniquely identifies the device.

Use this node with the Device Profile Save node to create a trusted profile from the collected data. You can use the trusted device profile in subsequent authentication attempts; for example, with the Device Match node and Device Location Match node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

Single outcome path.

Properties

Property	Usage
Maximum Profile Size (KB)	Specifies the maximum accepted size, in kilobytes, of a device profile. If the collected profile data exceeds this size, authentication fails. Default: `3`
Collect Device Metadata	Specifies whether device metadata is requested.
Collect Device Location	Specifies whether device location is requested.

Property	Usage
Message	Specifies an optional message to display to the user while the node collects the requested data. You can provide the message in multiple languages by specifying the locale in the `KEY` field; for example, `en-US`. The locale selected for display is based on the user's locale settings in their browser. Messages provided in the node override the defaults provided by AM.

Device Profile Save node

Persists collected device data to a user's profile in the identity store.

Use this node with the Device Profile Collector node to reuse the collected data in future authentications; for example, with the Device Match node and Device Location Match node.

You must establish the identity of the user before attempting to save to their profile.

A user profile can contain multiple device profiles. Use the **Maximum Saved Profiles** property to configure the maximum number of device profiles to persist per user. Saving a device profile with the same identifier as an existing entry overwrites the original record, and does not increment the device profile count.

In a Ping Identity Platform deployment, the end user UI displays saved device profiles to end users.

In an AM standalone deployment, the PingAM UI **does not** display saved device profiles to end users.

You can manage device profiles over REST, by using the `/json/users/user/devices/profile` endpoint for the realm.

Use the AM API Explorer for detailed information about the parameters supported by the `/devices/profile` endpoint and to test it against your deployed AM instance.

In the AM admin UI, select the **Help** icon, and then go to **API Explorer** > **/users** > **/{user}** > **/devices** > **/profile**.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

Single outcome path.

Properties

Property	Usage
Device Name Variable	Specifies the name of a variable in the shared node state that contains an alias label for the device profile.
Maximum Saved Profiles	Specify the maximum number of device profiles to save in a user's profile. When the maximum is reached, saving a new profile replaces the least-recently used profile.
Save Device Metadata	Specifies whether device metadata is saved to the user's profile.
Save Device Location	Specifies whether device location metadata is saved to the user's profile.

Device Tampering Verification node

Specifies a threshold for deciding if the device has been tampered with; for example, if it has been rooted or jailbroken.

The device scores between zero and one, based on the likelihood that is has been tampered with or may pose a security risk. For example, an emulator scores the maximum of `1`.

Use this node with the Device Profile Collector node to retrieve the tampering score from the device.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `Not Tampered`
- `Tampered`

Evaluation continues along the `Not Tampered` path if the device scores less than or equal to the configured threshold; otherwise, evaluation continues along the `Tampered` path.

Properties

Property	Usage
Score Threshold	Specifies the score threshold for determining if a device has been tampered with. Enter a decimal fraction, between `0` and `1`; for example, `0.75`. The higher the score returned from the device, the more likely the device is jailbroken, rooted, or is a potential security risk. Emulators score the maximum; `1`.

Persistent Cookie Decision node

The **Persistent Cookie Decision** node checks for the existence of a specified persistent cookie (default: `session-jwt`).

If the cookie is present, the node verifies the signature of the JWT stored in the cookie with the configured signing key.

If the configured signing key isn't valid, AM checks the signature against all valid signing keys mapped to the configured secret label.

If the signature is valid, the node decrypts the payload of the JWT using the key pair defined in the active secret mapped to the `am.authentication.nodes.persistentcookie.encryption` secret label.

If there isn't a valid secret label mapping in a secret store, AM uses the key pair specified in **Realms** > ***Realm Name*** > **Authentication** > **Settings** > **Security** > **Persistent Cookie Encryption Certificate Alias**. The global setting is found under **Configure** > **Authentication** > **Core Attributes** > **Security**.

The decrypted JSON payload includes information, such as the UID of the identity and the client IP address. Enable **Enforce Client IP** to verify that the current IP address and the client IP address in the cookie are the same.

Note

This node recreates the specified persistent cookie, updating the value for the idle time property and the JWT `kid` header with the stable ID used to sign the JWT.
Therefore, the node has cookie creation properties similar to the Set Persistent Cookie node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node requires the `realm` property, which AM sets by default.

Dependencies

To authenticate successfully, the tree must have set a persistent cookie using a node such as the Set Persistent Cookie node.

Configuration

Property	Usage
Idle Timeout	The maximum idle time allowed before the persistent cookie is invalidated, in hours. If no requests are received and the time is exceeded, the cookie is no longer valid.
Enforce Client IP	When enabled, ensures that the persistent cookie is only used from the same client IP to which the cookie was issued.
Use Secure Cookie	When enabled, adds the `Secure` flag to the persistent cookie. If the `Secure` flag is included, the cookie can only be transferred over HTTPS. When a request is made over HTTP, the cookie is not made available to the application.
Use HTTP Only Cookie	When enabled, adds the `HttpOnly` flag to the persistent cookie. When the `HttpOnly` flag is included, that cookie will not be accessible through JavaScript. According to RFC 6265, the `HttpOnly` flag, "instructs the user agent to omit the cookie when providing access to cookies via 'non-HTTP' APIs (for example, a web browser API that exposes cookies to scripts)."

HMAC Signing Key	The key to use for HMAC signing of the persistent cookie.
	Note
	This property is *deprecated*. Use the **HMAC Signing Key Secret Label Identifier** instead.
	If you set an **HMAC Signing Key Secret Label Identifier**, this signing key is ignored.
	Values must be base64-encoded and at least 256 bits (32 bytes) long.
	To generate an HMAC signing key, run one of the following commands:
	`$ openssl rand -base64 32`
	or
	`$ cat /dev/urandom | LC_ALL=C tr -dc 'a-zA-Z0-9' | fold -w 32 | head -n 1|base64`

Property	Usage
HMAC Signing Key Secret Label Identifier	An identifier used to create a *secret label* for mapping to a secret in a secret store. AM uses this identifier to create a specific secret label for the signing key for this node. The secret label takes the form `am.authentication.nodes.persistentcookie.identifier.signing` where identifier is the value of **HMAC Signing Key Secret Label Identifier**. The identifier can only contain alphanumeric characters `a-z`, `A-Z`, `0-9`, and periods (`.`). It can't start or end with a period. If you set an **HMAC Signing Key Secret Label Identifier** and AM finds a matching secret in a secret store, the **HMAC Signing Key** is ignored. If **HMAC Signing Key** is empty, AM uses the value configured for `am.default.authentication.nodes.persistentcookie.signing` for the realm, or at the global level if undefined. For greater security, you should rotate signing keys periodically. When you rotate a key, update the corresponding mapping in the realm secret store configuration to reflect this identifier. **Important** To read the persistent cookies generated by the Set Persistent Cookie node, ensure the nodes use the same HMAC signing key.
Persistent cookie name	The name of the persistent cookie to check.

Outputs

The node copies shared state into the outgoing node state. It records the user identity and stores the cookie name as a session property.

The node adds the `UpdatePersistentCookieTreeHook`, which runs when the tree completes.

Outcomes

- `True`
- `False`

Evaluation continues along the `True` outcome path if the persistent cookie is present and all the verification checks are satisfied; otherwise, evaluation continues along the `False` outcome path.

Errors

The node logs the following warning messages:

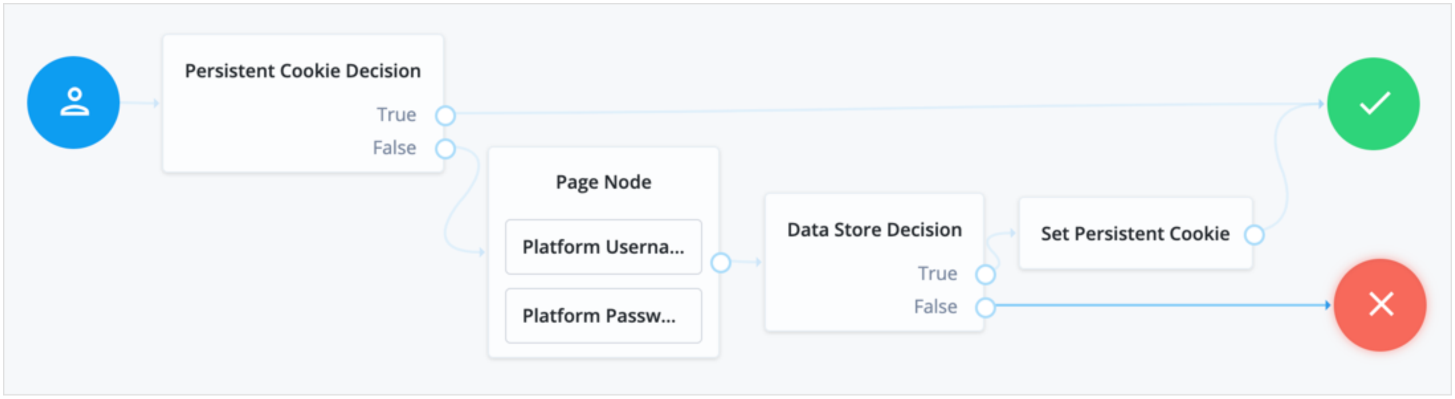
- `Attempt to verify JWT failed, attempting other valid keys`
- `Failed to parse universal id username from claim openam.usr`

The node logs the following error messages:

- `Claims context not found`
- `Failed to find signing key with associated keyID`
- `jwt reconstruction error`
- `Authentication failed. Jwt claim Realm does not match`
- `Authentication failed. Cannot read the user from null claims`
- `Authentication failed. Cannot read the user from empty claims`
- `Failed to parse universal Id from claim: openam.usr`
- `Authentication failed. Client IP is different`

Example

The following example authenticates the user based on a persistent cookie, if possible:



Set Custom Cookie node

The **Set Custom Cookie** node lets you store a custom cookie on the client in addition to the session cookie.

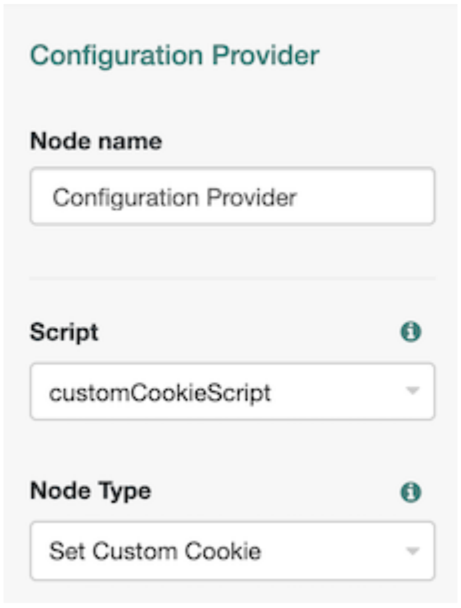
The node uses the specified properties to create a cookie with a custom name and value. It can also set attributes, such as the cookie path, domain, expiry, and security flags.

Use this node with the Configuration Provider node to extend custom capabilities. For example, create a `Config Provider` script to set custom static values or access values from the shared node state.

Include all the attributes in the configuration provider script's `config` map. The following example sets the attributes of the custom cookie to static values:

```
config = {
    "name": "testname",
    "value": "testvalue",
    "maxAge": "60",
    "domain": "am.example.com",
    "path": "/",
    "useSecureCookie": false,
    "useHttpOnlyCookie": false,
    "sameSite": "LAX"
};
```

Reference the script when you create a Configuration Provider node, and set the **Node Type** to `Set Custom Cookie`:



Configuration Provider

Node name: Configuration Provider

Script: customCookieScript

Node Type: Set Custom Cookie

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node reads the user data from the shared node state.

It requires a predecessor node that gathers the user data.

Configuration

Property	Usage
Custom Cookie Name *(required)*	The name of the custom cookie. The cookie name can contain any US-ASCII characters except for: space, tab, control, or a separator character (`()<>@,;:"/[]?=\{}`).
Custom Cookie Value *(required)*	The value of the custom cookie.
Max Age	The length of time the custom cookie remains valid, in seconds. If that time is exceeded, the cookie is no longer valid. AM sets the `Max-Age` and `Expires` attributes in the cookie to increase compatibility with different browsers. If omitted, the cookie expires at the end of the current session. The precise implementation of this is determined by the specific browser. Refer to RFC 6265 for details.
Custom Cookie Domain	The domain the custom cookie will be sent to. If you specify a value here, AM sets a domain cookie. For example, if you set this property to `am.example.com`, AM sets a cookie on `.am.example.com`. Note the leading `.` indicating a domain cookie rather than a host cookie. If you don't set a value here, AM sets a host level cookie on the FQDN on which the client accessed AM. For example, if the client accesses AM at `https://am.example.com` and this property is empty, AM sets a host cookie on `am.example.com`.
Custom Cookie Path	The path of the custom cookie.
Use Secure Cookie	When enabled, adds the `Secure` flag to the custom cookie. If you include the `Secure` flag, the cookie can only be transferred over HTTPS. When a request is made over HTTP, the cookie isn't made available to the application.
Use HTTP Only Cookie	When enabled, adds the `HttpOnly` flag to the custom cookie. If you include the `HttpOnly` flag, the cookie isn't accessible to scripts.
Custom Cookie SameSite attribute	Sets the `SameSite` attribute of the custom cookie. The default value is `LAX`, to align with most modern browsers. Learn more in SameSite cookie rules.

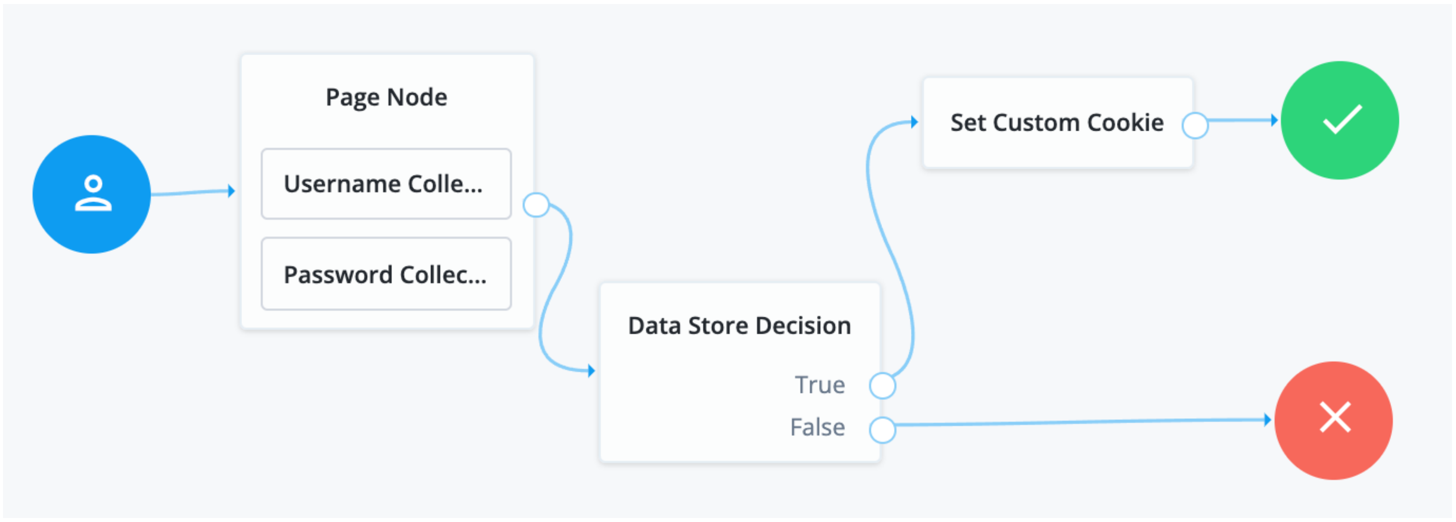
Outcomes

Single outcome path.

The cookie is created when AM next returns to the client.

Example

This example uses this node in a login flow. The node sets the custom cookie in the client browser after the user has successfully authenticated:



Set Persistent Cookie node

Creates the specified persistent cookie, the default being `session-jwt`.

The cookie contains a JWT with a JSON payload including information such as the UID of the identity, and the client IP address.

The node encrypts the payload of the JWT using the key pair defined in the active secret mapped to the `am.authentication.nodes.persistentcookie.encryption` secret label.

If there isn't a valid secret label mapping in a secret store, AM uses the key pair specified in **Realms** > ***Realm Name*** > **Authentication** > **Settings** > **Security** > **Persistent Cookie Encryption Certificate Alias**. The global setting is found under **Configure** > **Authentication** > **Core Attributes** > **Security**.

The node signs the cookie with the HMAC signing key defined in the node properties or the secret store with the mapped secret label. Configure nodes that read the persistent cookie, for example Persistent Cookie Decision node, with the same HMAC signing key.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

When the authentication tree completes successfully, the `CreatePersistentCookieTreeHook` treehook for this node uses session properties to create the persistent cookie.

Dependencies

A secret store configured for storing the dynamic secret label mapping to the cookie's signing key.

Configuration

Property	Usage
Idle Timeout	The maximum amount of idle time allowed before the persistent cookie is invalidated, in hours. If no requests are received before the timeout, the cookie is no longer valid.
Max life	The length of time the persistent cookie remains valid, in hours. After this time has passed, the cookie is no longer valid.
Use Secure Cookie	When enabled, adds the `Secure` flag to the persistent cookie. If the `Secure` flag is included, the cookie can only be transferred over HTTPS. When a request is made over HTTP, the cookie is not made available to the application.
Use HTTP Only Cookie	When enabled, adds the `HttpOnly` flag to the persistent cookie. When the `HttpOnly` flag is included, that cookie will not be accessible through JavaScript. According to RFC 6265, the `HttpOnly` flag, "instructs the user agent to omit the cookie when providing access to cookies via 'non-HTTP' APIs (for example, a web browser API that exposes cookies to scripts)."

HMAC Signing Key	A key to use for HMAC signing of the persistent cookie.
	Note
	This property is *deprecated*. Use the **HMAC Signing Key Secret Label Identifier** instead. If you set an **HMAC Signing Key Secret Label Identifier**, this signing key is ignored.
	Values must be base64-encoded and at least 256 bits (32 bytes) long.
	To generate an HMAC signing key, run one of the following commands:
	`$ openssl rand -base64 32`
	or
	`$ cat /dev/urandom | LC_ALL=C tr -dc 'a-zA-Z0-9' | fold -w 32 | head -n 1|base64`

Property	Usage
HMAC Signing Key Secret Label Identifier	An identifier used to create a *secret label* for mapping to a secret in a secret store. AM uses this identifier to create a specific secret label for the signing key for this node. The secret label takes the form `am.authentication.nodes.persistentcookie.identifier.signing` where identifier is the value of **HMAC Signing Key Secret Label Identifier**. The identifier can only contain alphanumeric characters `a-z`, `A-Z`, `0-9`, and periods (`.`). It can't start or end with a period. If you set an **HMAC Signing Key Secret Label Identifier** and AM finds a matching secret in a secret store, the **HMAC Signing Key** is ignored. If **HMAC Signing Key** is empty, AM uses the value configured for `am.default.authentication.nodes.persistentcookie.signing` for the realm, or at the global level if undefined. For greater security, you should rotate signing keys periodically. When you rotate a key, update the corresponding mapping in the realm secret store configuration to reflect this identifier. **Important** To read the persistent cookies this node generates, ensure the nodes use the same HMAC signing key.
Persistent Cookie Name	The name used for the persistent cookie.

Outputs

The node stores the cookie name in the session properties.

The node adds the `CreatePersistentCookieTreeHook` treehook, which runs when the tree completes.

Outcomes

Single outcome path.

Errors

The node logs the following warning messages:

- `Unable to create signing key from provided configuration.`

The node logs the following error messages:

- `Tree hook creation exception`
- `No signing keys available to sign JWT`
- `Error creating jwt string`

Example

Refer to the Persistent Cookie Decision node example.

Federation nodes

OAuth 2.0 node

Lets AM authenticate users of OAuth 2.0-compliant resource servers.

References in this section are to RFC 6749, The OAuth 2.0 Authorization Framework.

Note

This node and its related services, are deprecated.
You can find information on the new methods for implementing social authentication in Social authentication.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	No

Outcomes

- `Account Exists`
- `No account Exists`

Evaluation continues along the `Account Exists` path if an account matching the attributes retrieved from the social identity provider is found in the user data store; otherwise, evaluation continues along the `No account exists` path.

Properties

Property	Usage
Client ID *(required)*	Specifies the `client_id` parameter as described in section 2.2 of The OAuth 2.0 Authorization Framework (RFC 6749).
Client Secret *(required)*	Specifies the `client_secret` parameter as described in section 2.3 of The OAuth 2.0 Authorization Framework (RFC 6749).

Property	Usage
Authentication Endpoint URL *(required)*	Specifies the URL to the social provider's endpoint handling authentication as described in section 3.1 of The OAuth 2.0 Authorization Framework (RFC 6749). Example: `https://accounts.google.com/o/oauth2/v2/auth`
Access Token Endpoint URL *(required)*	Specifies the URL to the endpoint handling access tokens as described in section 3.2 of The OAuth 2.0 Authorization Framework (RFC 6749). Example: `https://www.googleapis.com/oauth2/v4/token`
User Profile Service URL *(required)*	Specifies the user profile URL that returns profile information. Example: `https://www.googleapis.com/oauth2/v3/userinfo`
OAuth Scope *(required)*	Specifies a list of user profile attributes that the client application requires, according to The OAuth 2.0 Authorization Framework (RFC 6749). Ensure you use the correct scope delimiter required by the identity provider, including commas or spaces. The list depends on the permissions that the resource owner, such as the end user, grants to the client application.
Scope Delimiter *(required)*	Specifies the delimiter used to separate scope values. Some authorization servers use non-standard separators for scopes, for example commas.
Redirect URL *(required)*	Specifies the URL the user is redirected to by the social identity provider after authenticating. For authentication trees in AM, set this property to the URL of the UI. For example, `https://am.example.com:8443/am/XUI/`.
Social Provider *(required)*	Specifies the name of the social provider for which this module is being set up. Example: `Google`
Auth ID Key *(required)*	Specifies the attribute the social identity provider uses to identify an authenticated individual. Example: `id`
Use Basic Auth	Specifies that the client uses HTTP Basic authentication when authenticating to the social provider. Default: `true`
Account Provider *(required)*	Specifies the name of the class that implements the account provider. Default: `org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider`

Property	Usage

Account Mapper *(required)*	Specifies the name of the class that implements the method of locating local accounts based on the attributes returned from the social identity provider.
	Provided implementations are:
	`org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper`
	The Account Mapper classes can take two constructor parameters:
	1. A comma-separated list of attributes
	2. A prefix to apply to their values.
	For example, to prefix all received property values with `facebook-` before searching, specify:
	`org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper|*|facebook-`

Attribute Mapper *(required)*	Specifies the list of fully qualified class names for implementations that map attributes from the OAuth 2.0 authorization server to AM profile attributes.
	Provided implementations are:
	`org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper`
	The Attribute Mapper classes can take two constructor parameters to help differentiate between the providers:
	1. A comma-separated list of attributes
	2. A prefix to apply to their values.
	For example, to prefix all incoming values with `facebook-`, specify:
	`org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper|*|facebook-`
	To prefix all incoming values use an asterisk (`*`) as the attribute list. This prefixes all values, including email addresses, postal addresses, and so on.

Property	Usage

Account Mapper Configuration	Specifies the attribute configuration used to map the account of the user authenticated in the OAuth 2.0 provider to the local data store in AM.
	Valid values are in the form `provider-attr=local-attr`.
	Examples:
	`email=mail`
	`id=facebook-id`
	Tip
	When using the `org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper` class, you can parse JSON objects in mappings using dot notation.
	For example, given a JSON payload of:
	`{ "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } }`
	You can create a mapper, such as `name.first_name=cn`.

Property	Usage

Attribute Mapper Configuration	Map of OAuth 2.0 provider user account attributes to local user profile attributes, with values in the form `provider-attr=local-attr`.
	Examples:
	`first_name=givenname`
	`last_name=sn`
	`name=cn`
	`email=mail`
	`id=facebook-id`
	`first_name=facebook-fname`
	`last_name=facebook-lname`
	`email=facebook-email`
	Tip
	When using the `org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper` class, you can parse JSON objects in mappings using dot notation.
	For example, given a JSON payload of:
	`{ "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } }`
	You can create a mapper, such as `name.first_name=cn`.

Save attributes in the session	When enabled, saves the attributes in the Attribute Mapper Configuration field to the AM session.
OAuth 2.0 Mix-Up Mitigation Enabled	Controls whether the OAuth 2.0 authentication node carries out additional verification steps when it receives the authorization code from the authorization server. Specifies that the client must compare the issuer identifier of the authorization server upon registration with the issuer value returned as the `iss` response parameter. If they do not match, the client must abort the authorization process. The client must also confirm that the authorization server's response is intended for the client by comparing the client's client identifier to the value of the `client_id` response parameter. When this is enabled, set the Token Issuer property so that the validation can succeed. The authorization code response contains an issuer value (`iss`) for the client to validate. **Note** Refer to the authorization server's documentation for the value it uses for the issuer field. Learn more in section 4 of OAuth 2.0 Mix-Up Mitigation Draft.
Token Issuer	Corresponds to the expected issuer identifier value in the `iss` field of the ID token. Example: `https://accounts.google.com`

OpenID Connect node

Lets AM authenticate users of OpenID Connect-compliant resource servers.

As OpenID Connect is an additional layer on top of OAuth 2.0, described in RFC 6749, The OAuth 2.0 Authorization Framework. OpenID Connect is described in the OpenID Connect Core 1.0 incorporating errata set 1 specification.

Note

This node and its related services, are deprecated.
You can find information on the new methods for implementing social authentication in Social authentication.

The OpenID Connect node implements the Authorization code grant.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	No

Outcomes

- `Account Exists`
- `No account Exists`

Evaluation continues along the `Account Exists` path if an account matching the attributes retrieved from the OpenID Connect identity provider is found in the identity store; otherwise, evaluation continues along the `No account exists` path.

Properties

Property	Usage
Client ID *(required)*	Specifies the `client_id` parameter as described in section 2.2 of The OAuth 2.0 Authorization Framework (RFC 6749).
Client Secret *(required)*	Specifies the `client_secret` parameter as described in section 2.3 of The OAuth 2.0 Authorization Framework (RFC 6749).
Authentication Endpoint URL *(required)*	Specifies the URL to the social provider's endpoint handling authentication as described in section 3.1 of The OAuth 2.0 Authorization Framework (RFC 6749). Example: `https://accounts.google.com/o/oauth2/v2/auth`

Property	Usage
Access Token Endpoint URL *(required)*	Specifies the URL to the endpoint handling access tokens as described in section 3.2 of The OAuth 2.0 Authorization Framework (RFC 6749). Example: `https://www.googleapis.com/oauth2/v4/token`
User Profile Service URL *(required)*	Specifies the user profile URL that returns profile information. If not specified, attributes are mapped from the claims returned by the `id_token`, and no call to a user profile endpoint is made. Example: `https://www.googleapis.com/oauth2/v3/userinfo`
OAuth Scope	Specifies a list of user profile attributes that the client application requires, according to The OAuth 2.0 Authorization Framework (RFC 6749). Ensure you use the correct scope delimiter required by the identity provider, including commas or spaces. The list depends on the permissions that the resource owner, such as the end user, grants to the client application.
Redirect URL	Specifies the URL the user is redirected to by the social identity provider after authenticating. For authentication trees in AM, set this property to the URL of the UI. For example, `https://am.example.com:8443/am/XUI/`.
Social Provider *(required)*	Specifies the name of the OpenID Connect provider for which this node is being set up. Example: `Google`
Auth ID Key	Specifies the attribute the social identity provider uses to identify an authenticated individual. Example: `sub`
Use Basic Auth	Specifies that the client uses HTTP Basic authentication when authenticating to the social provider. Default: `true`
Account Provider	Specifies the name of the class that implements the account provider. Default: `org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider`

Property	Usage
Account Mapper	Specifies the name of the class that implements the method of locating local accounts based on the attributes returned from the social identity provider. The provided implementations is `org.forgerock.openam.authentication.modules.oidc.JwtAttributeMapper`. The Account Mapper classes can take two constructor parameters: 1. A comma-separated list of attributes 2. A prefix to apply to their values. For example, to prefix all received property values with `openid-` before searching, specify: `org.forgerock.openam.authentication.modules.oidc.JwtAttributeMapper|*|openid-`
Attribute Mapper	Specifies the list of fully qualified class names for implementations that map attributes from the authorization server to AM profile attributes. The provided implementations is `org.forgerock.openam.authentication.modules.oidc.JwtAttributeMapper`. The Attribute Mapper classes can take two constructor parameters to help differentiate between the providers: 1. A comma-separated list of attributes 2. A prefix to apply to their values. For example, to prefix incoming `iplanet-am-user-alias-list` values with `openid-`, specify: `org.forgerock.openam.authentication.modules.oidc.JwtAttributeMapper`
iplanet-am-user-alias-list	openid- To prefix all incoming values use an asterisk (`*`) as the attribute list. This prefixes all values, including email addresses, postal addresses, and so on.
Account Mapper Configuration	Specifies the attribute configuration used to map the account of the user authenticated in the provider to the local identity store in AM. To add a mapping, specify the name of the provider attribute as the key, and the local attribute to map to as the value. For example, click **Add**, then specify `sub` in the **Key** field and `iplanet-am-user-alias-list` in the **Value** field, and click **+**.

Property	Usage

Attribute Mapper Configuration	Specifies how to map provider user attributes to local user profile attributes.
	To add a mapping, specify the name of the provider attribute as the Key, and the local attribute to map to as the Value.
	For example, click **Add**, then specify `id` in the **Key** field and `facebook-id` in the **Value** field, and click **+**.
	Examples:
	`first_name=givenname`
	`last_name=sn`
	`name=cn`
	`email=mail`
	`id=facebook-id`
	`first_name=facebook-fname`
	`last_name=facebook-lname`
	`email=facebook-email`

Save attributes in the session	When enabled, saves the attributes in the Attribute Mapper Configuration field to the AM session.
OAuth 2.0 Mix-Up Mitigation Enabled	Controls whether the authentication node carries out additional verification steps when it receives the authorization code from the authorization server. Specifies that the client must compare the issuer identifier of the authorization server upon registration with the issuer value returned as the `iss` response parameter. If they do not match, the client must abort the authorization process. The client must also confirm that the authorization server's response is intended for the client by comparing the client's client identifier to the value of the `client_id` response parameter. When this is enabled, set the Token Issuer property so that the validation can succeed. The authorization code response contains an issuer value (`iss`) for the client to validate. **Note** Refer to the authorization server's documentation for the value it uses for the issuer field. Learn more in section 4 of OAuth 2.0 Mix-Up Mitigation Draft.
Token Issuer *(required)*	Corresponds to the expected issuer identifier value in the `iss` field of the ID token. Example: `https://accounts.google.com`

Property	Usage
OpenID Connect Validation Type *(required)*	Specifies how to validate the ID token received from the OpenID Connect provider. This ignores keys specified in JWT headers, such as `jku` and `jwe`. The following options are available to validate an incoming OpenID Connect ID token: **`Well Known URL`** ***(Default)*** Retrieves the provider's keys based on the information provided in its OpenID Connect configuration URL. Specify the provider's configuration URL in the OpenID Connect Validation Value field; for example, `https://accounts.google.com/.well-known/openid-configuration`. **`Client Secret`** Validates the ID token signature with a specified client secret key. Specify the key to use in the OpenID Connect Validation Value field. **`JWK URL`** Retrieve the necessary JSON web key from the URL that you specify. Specify the provider's JWK URI in the OpenID Connect Validation Value field; for example, `https://www.googleapis.com/oauth2/v3/certs`.
OpenID Connect Validation Value	Provide the URL or secret key used to verify an incoming ID token, depending on the value selected in the OpenID Connect Validation Type property.

OIDC ID Token Validator node

The **OIDC ID Token Validator** node lets AM rely on an OIDC provider (OP)'s ID token to authenticate an end user. The node evaluates whether the ID token is valid according to the OIDC specification.

To configure the node, first get an `id_token` from an OIDC client and examine the decoded JWT to view the required claims values.

This example uses an `id_token` from the OAuth 2.0 Playground:

```
{
    "iss": "https://accounts.google.com",
    "azp": "407408718192.apps.googleusercontent.com",
    "aud": "407408718192.apps.googleusercontent.com",
    "sub": "111730983950574648607",
    "at_hash": "kvQJZrGcnNMZqM4w68DFBA",
    "iat": 1677608448,
    "exp": 1677612048
}
```

The `iss`, `azp`, and `aud` claims provide the values for the node's `Token Issuer`, `Authorized parties` and `Audience name` properties respectively.

To use the OIDC ID Token Validator node to authenticate a user, first configure the node to run a transformation script that maps the user attributes from the JWT to local attributes. You can then create a journey with a Scripted Decision node that stores the attributes in the shared node state so that you can authenticate the user with an ID token.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

None.

Dependencies

A valid OIDC ID token provided in the HTTP request header.

Configuration

Property	Usage
OpenID Connect Validation Type	To validate the ID token from the OP, the node requires either a URL to get the public keys for the provider, or the symmetric key for an ID token signed with an HMAC-based algorithm. Select one of the following options to determine how the node retrieves the required information: • `Well Known URL` (default): Validate with the keys specified in the OP's `/.well-known/openid-configuration` JSON document. • `Client Secret`: Validate the ID token signature with the provided client secret. • `JWK URL`: Validate with the keys retrieved from the URL to the OP's JSON Web Key Set (JWKS).
OpenID Connect Validation Value	The well-known URL or URL to the JWK location, depending on the value of `OpenID Connect Validation Type`. If the validation type is `Client Secret`, this value is ignored and the `Client Secret Label` is used instead. For example: `https://accounts.google.com/.well-known/openid-configuration`

Property	Usage
Client Secret Label	The secret label to which the OIDC client secret should be mapped. Only required if the validation type is `Client Secret`. You can specify an existing label or AM creates a new one dynamically. The label can only contain alphanumeric characters `a-z`, `A-Z`, `0-9`, and periods (`.`). It can't start or end with a period.
ID Token Header Name	The name of the HTTP request header referencing the ID token. Default: `oidc_id_token`
Token Issuer	The issuer of the OIDC ID token, which is checked against the `iss` claim in the ID token. For example: `https://accounts.google.com`
Audience name	The case-sensitive name of the intended audience for this node, which is checked against the `aud` claim in the ID token.
Authorized parties	The authorized parties from which the node accepts ID tokens, which is checked against the `azp` claim in the ID token. The value can be either a case-sensitive string or a URI.
Transformation Script	Select a `Social Identity Provider Profile Transformation` script that maps ID token attributes to local attributes. Find examples of transformation (normalization) scripts in the Example or the `*-profile-normalization.js` scripts in Sample scripts.
Script Inputs	A list of state inputs for the script. Default: `*`
Unreasonable Lifetime Limit	Specify the maximum permitted lifetime of the token in minutes. If the `iat` claim is present, the token must expire within the specified duration. Default: `60`

Outputs

The node relies on the transformation script to set profile attributes required later in the journey.

Outcomes

- `True`
- `False`

Evaluation continues along the `True` path if the ID token is valid; otherwise, evaluation continues along the `False` path.

Errors

The node logs the following warnings:

- `No OpenIdConnect ID Token referenced by header value: {}` : if the node can't read the ID token in the HTTP request header.
- `Error evaluating the script` : if there's a problem with the transformation script.

The node logs an error if an `AuthLoginException` occurs during node processing.

Example

This example demonstrates how to use the OIDC ID Token Validator node as part of a journey to validate an ID token and authenticate the user.

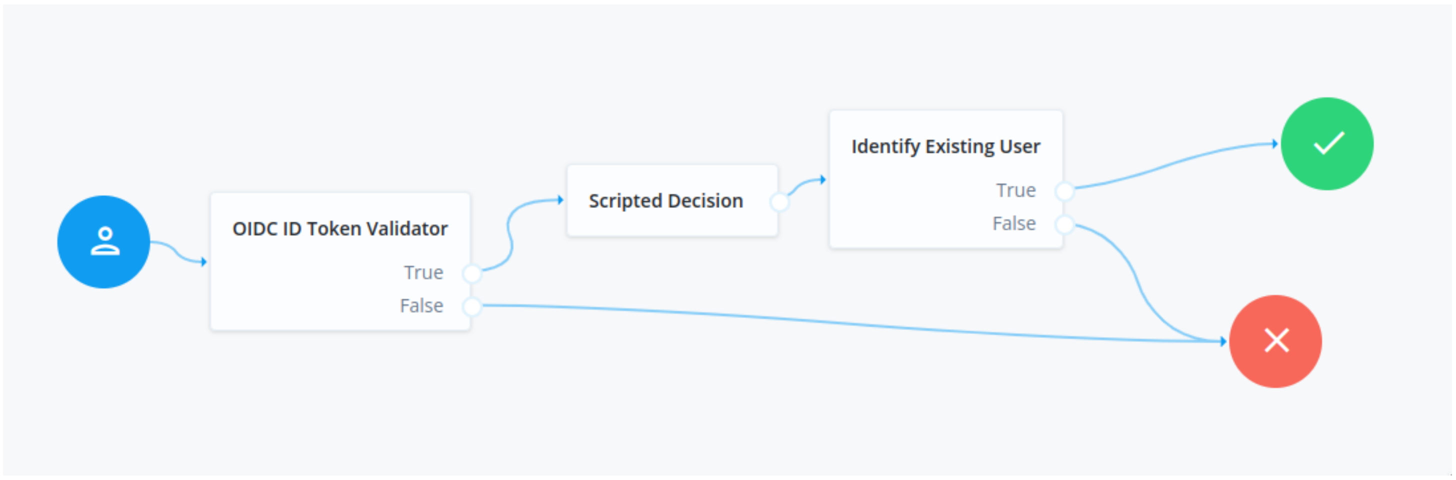


Figure 1. Ping Identity Platform deployment

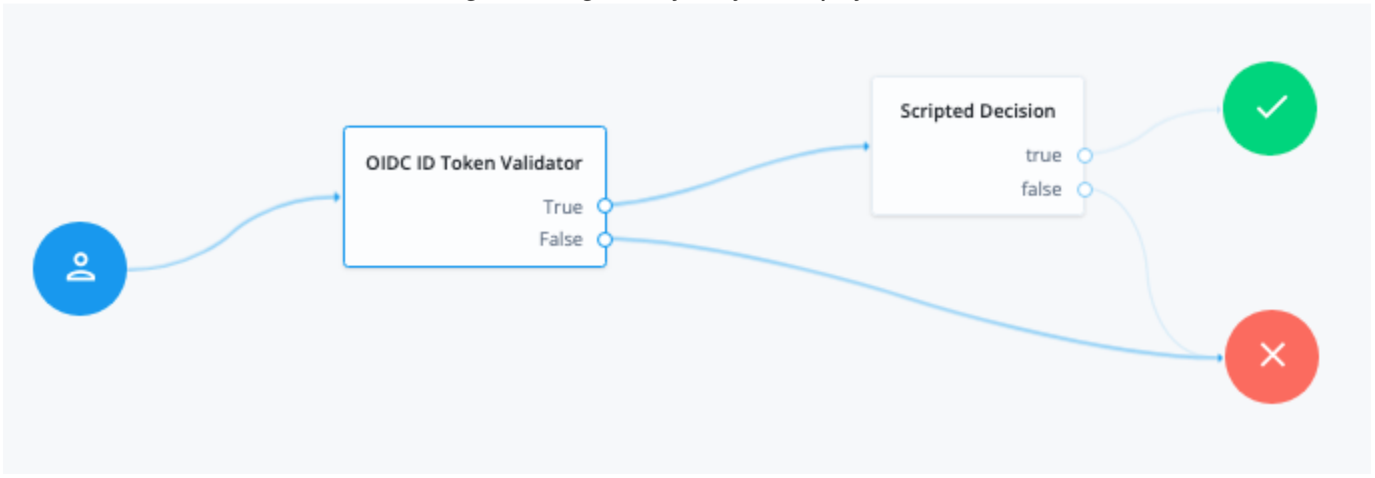


Figure 2. PingAM deployment

You can access all the provided JWT claims through the `jwtClaims` attribute. This JavaScript, configured as the node's transformation script, retrieves the user ID from the JWT.

```
(function () {
    var fr = JavaImporter( org.forgerock.json.JsonValue);

    var identity = fr.JsonValue.json(fr.JsonValue.object());
    identity.put('uid', jwtClaims.get('sub'));

    return identity;
}());
```

Ping Identity Platform

A Scripted Decision node runs this script to find the username from `lookupAttributes` and store it in the shared node state:

Next-generation

```
var attributes = nodeState.get("lookupAttributes");
var userName = attributes.get("uid");

// Add userName in objectAttributes to nodeState for use by Identify Existing User node.
nodeState.putShared("objectAttributes", attributes);

// Add username at the root level so that a session can be created
nodeState.putShared("username", userName);

action.goTo('true');
```

Legacy

```
var attributes = nodeState.get("lookupAttributes");
var userName = attributes.get("uid").asString();

// Add userName in objectAttributes to nodeState for use by Identify Existing User node.
nodeState.putShared("objectAttributes", attributes);

// Add username at the root level so that a session can be created
nodeState.putShared("username", userName);

outcome = "true";
```

The Identify Existing User node then performs a lookup on IDM with the `_id` attribute using the username saved into shared state by the Scripted Decision node.

The following REST call authenticates the user with the example journey, providing the ID token in the header:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=2.0, protocol=1.0" \
--header "oidc_id_token: <id_token>" \
"https://<tenant-env-fqdn>/am/json/realms/root/realms/alpha/authenticate?
authIndexType=service&authIndexValue=myJourney"
{
    "tokenId": "AQIC5w…NTcy*",
    "successUrl": "/openam/console",
    "realm": "/alpha"
}
```

Standalone PingAM

A Scripted Decision node runs this script to find the user ID from `lookupAttributes` and store it in the shared node state:

Next-generation

```
var attributeName = "uid";
var attributes = nodeState.get("lookupAttributes");
var uid = attributes.get(attributeName);

// get the identity for the sub claim (stored as uid)
var identity = idRepository.getIdentity(uid);

// verify the identity exists
var userName = identity.getAttributeValues(attributeName);

if (!userName.isEmpty()) {
    nodeState.putShared("username", userName[0]);
    action.goTo('true');
} else {
    action.goTo('false');
}
```

Legacy

```
var attributeName = "uid";
var attributes = nodeState.get("lookupAttributes");
var userName = attributes.get(attributeName).asString();
var identity = idRepository.getAttribute(userName, attributeName);

if (!identity.isEmpty()) {
    nodeState.putShared("username", identity.iterator().next());
    outcome = "true";
} else {
    outcome = "false";
}
```

The following REST call authenticates the user with the example tree, providing the ID token in the header:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=2.0, protocol=1.0" \
--header "oidc_id_token: <id_token>" \
"https://openam.example.com:8443/openam/json/realms/root/authenticate?
authIndexType=service&authIndexValue=myJourney"
{
    "tokenId": "AQIC5w…NTcy*",
    "successUrl": "/openam/console",
    "realm": "/alpha"
}
```

Provision Dynamic Account node

Provision an account following successful authentication by a SAML2 authentication node or the Social Provider Handler node.

Accounts are provisioned using properties defined in the attribute mapper configuration of a social authentication or SAML2 authentication node earlier in the flow.

If a password has been acquired from the user, for example, by using the Password Collector node, it is used when provisioning the account; otherwise, a 20 character random string is used.

In addition to retrieving the password from the node state, the Provision Dynamic Account node gets the `realm` value, and `attributes` and `userNames` from `userInfo` in the shared state. It sets the `username` attribute in the node's shared state.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	No

Outcomes

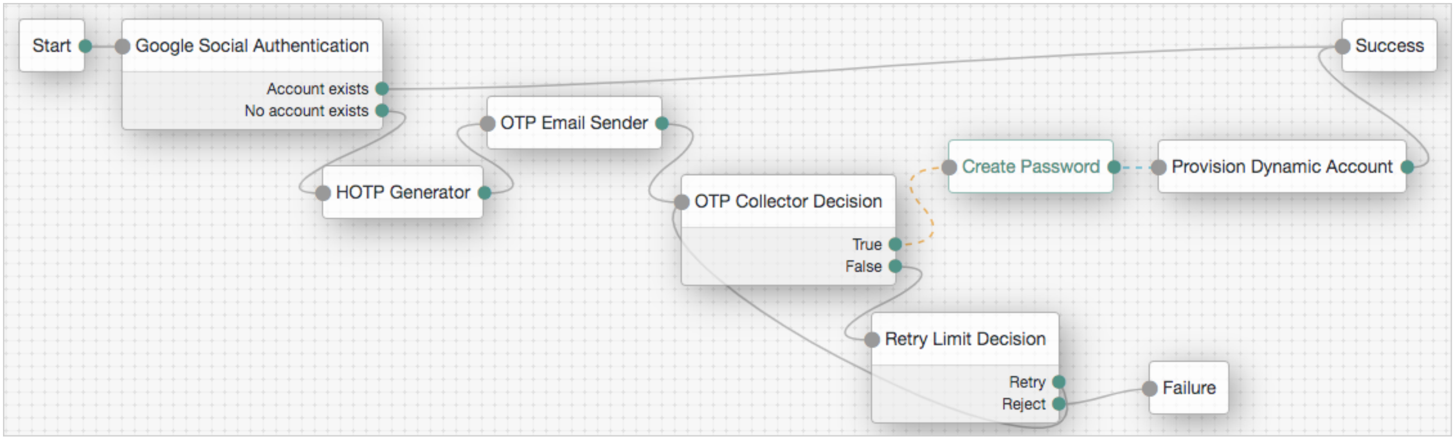
Single outcome path.

Properties

Property	Usage
Account Provider	Specifies the name of the class that implements the account provider. Default: `org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider`

Example

The following example uses this node to let users who have performed social authentication using Google provide a password and provision an account if they do not have a matching existing profile. They must enter a one-time passcode to verify they are the owner of the Google account.



Provision IDM Account node

Redirects users to an IDM instance to provision an account.

Note

This node and its related services, are deprecated.
You can find information on the new methods for implementing social authentication in Social authentication.

Ensure you configured the details of the IDM instance in AM, by navigating to **Configure > Global Services > IDM Provisioning**.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes

Product	Available?
Ping Identity Platform (self-managed)	No

Outcomes

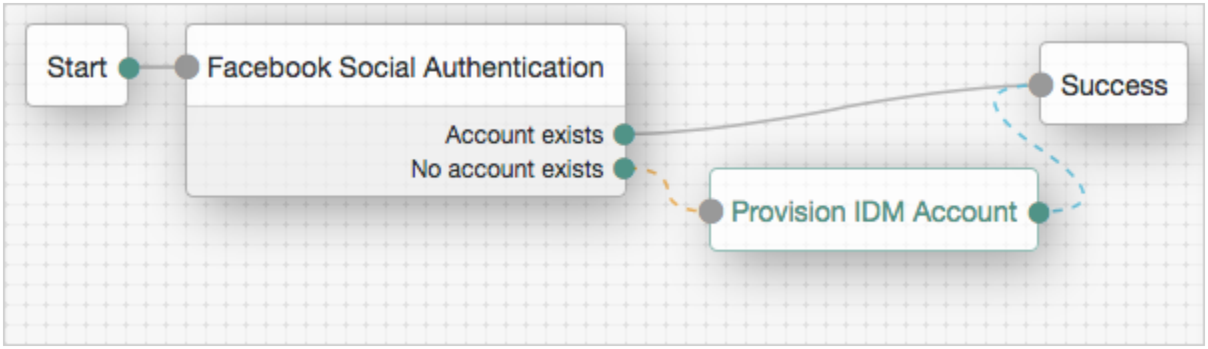
Single outcome path.

Properties

Property	Usage
Account Provider	Specifies the name of the class that implements the account provider. Default: `org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider`

Example

The following example uses this node to let users who have performed social authentication using Facebook provide a password and provision an account if they do not have a matching existing profile:



SAML2 Authentication node

Integrates SAML v2.0 SSO into an AM authentication flow.

Use this node when deploying SAML v2.0 single sign-on in integrated mode (SP-initiated SSO only).

Regardless of the outcome, `Account exists` or `No account exists`, if this node completes without failure, it sets the `successURL` parameter in the shared node state to the value of the `RelayState` parameter in the request. If the request does not provide a value for this parameter, the node uses the default `RelayState` value configured in the service provider (SP).

You can dynamically provision an account on the SP if it does not exist, or you can link the remote account to a local account using the Write Federation Information node.

Before attempting to configure a SAML2 authentication node, ensure that:

- You have configured a remote identity provider (IdP) and a hosted SP in a circle of trust in the same realm where the authentication node is configured.
- The service provider is configured for integrated mode.

  Learn more in SSO and SLO in integrated mode.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

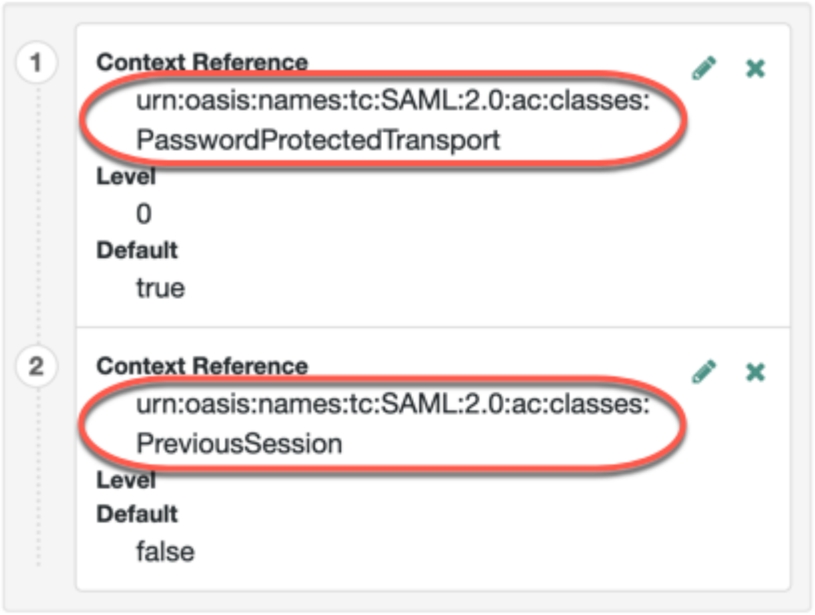
- `Account exists`
- `No account exists`

If a user account is found that matches the federated account, evaluation continues along the `Account exists` outcome; otherwise, evaluation continues along the `No account exists` outcome.

Properties

Property	Usage
IdP Entity ID	Specifies the name of the remote IdP.
SP MetaAlias	Specifies the local alias for the SP, in the format `/Realm Name/SP Name`.
Allow IdP to Create NameID	Specifies whether the IdP should create a new identifier for the authenticating user if none exists. For detailed information, refer to the section on the `AllowCreate` property in SAML Version 2.0 Errata 05. Default: `Enabled`

Property	Usage
Comparison Type	Specifies a comparison method to evaluate authentication context classes or statements. The value specified in this property overrides the value set in the SP configuration in AM admin UI under **Realms** > ***Realm Name*** > **Applications** > **Federation** > **Entity Providers** > ***Service Provider Name*** > **Assertion Content** > **Authentication Context** > **Comparison Type**. Valid comparison methods are `exact`, `minimum`, `maximum`, or `better`. For more information about the comparison methods, refer to the section on the `<RequestedAuthnContext>` element in Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Default: `minimum`

Authentication Context Class Reference	(Optional) Specifies one or more URIs for authentication context classes to be included in the SAML request.
	Authentication Context Classes are unique identifiers for an authentication mechanism. The SAML v2.0 protocol supports a standard set of authentication context classes, defined in Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0. In addition to the standard authentication context classes, you can specify customized authentication context classes.
	Any authentication context class you specify in this field must be supported for the service provider. In the AM admin UI, go to **Realms** > ***Realm Name*** > **Applications** > **Federation** > **Entity Providers** > ***Service Provider Name*** > **Assertion Content** > **Authentication Context**.
	
	1
	Context Reference
	urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
	Level
	0
	Default
	true
	2
	Context Reference
	urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession
	Level
	Default
	false
	When specifying multiple authentication context classes, use the `|` character to separate the classes. For example:
	`urn:oasis:names:tc:SAML:2.0:ac:classes:Password|`
	`urn:oasis:names:tc:SAML:2.0:ac:classes:TimesyncToken`

Property	Usage
Authentication Context Declaration Reference	(Optional) Specifies one or more URIs that identify authentication context declarations. When specifying multiple URIs, use the `|` character to separate the URIs. For more information, refer to the section on the `<RequestedAuthnContext>` element in Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0.
Request Binding	Specifies the format the SP will use to send the authentication request to the IdP. Valid values are `HTTP-Redirect` and `HTTP-POST`. Default: `HTTP-Redirect`
Response Binding	Specifies the format the IdP will use to send the response to the SP. Valid values are `HTTP-POST` and `HTTP-Artifact`. Default: `HTTP-Artifact`
Force IdP Authentication	Specifies whether the IdP forces authentication or if it can reuse existing security contexts. Default: Disabled
Passive Authentication	Specifies whether the IdP uses passive authentication or not. Passive authentication requires the IDP to only use authentication methods that do not require user interaction; for example, authenticating using an X.509 certificate. Default: Disabled

NameID Format	Specifies the SAML name ID format that will be requested in the SAML authentication request. For example:
	`urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`
	`urn:oasis:names:tc:SAML:2.0:nameid-format:transient`
	`urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`
	Default: `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`

Find examples in SSO and SLO in integrated mode.

Social Facebook node

Duplicates OAuth 2.0 node but is preconfigured to work with Facebook. You specify only the `Client ID` and `Client Secret`.

Note

This node and its related services, are deprecated.
You can find information on the new methods for implementing social authentication in Social authentication.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	No

Outcomes

- `Account exists`
- `No account exists`

Evaluation continues along the `Account Exists` path if an account matching the attributes retrieved from Facebook are found in the user data store; otherwise, evaluation continues along the `No account exists` path.

Properties

Property	Usage
Client ID	Specifies the `client_id` parameter as provided by Facebook.
Client Secret	Specifies the `client_secret` parameter as provided by Facebook.
Authentication Endpoint URL	Specifies the URL to the social provider's endpoint handling authentication as described in section 3.1 of The OAuth 2.0 Authorization Framework (RFC 6749). Default: `https://www.facebook.com/dialog/oauth`
Access Token Endpoint URL	Specifies the URL to the endpoint handling access tokens as described in section 3.2 of The OAuth 2.0 Authorization Framework (RFC 6749). Default: `https://graph.facebook.com/v2.12/oauth/access_token`
User Profile Service URL	Specifies the user profile URL that returns profile information. Default: `https://graph.facebook.com/v2.6/me?fields=name%2Cemail%2Cfirst_name%2Clast_name`
OAuth Scope	Specifies a comma-separated list of user profile attributes the client application requires, according to The OAuth 2.0 Authorization Framework (RFC 6749). The list depends on the permissions the resource owner, such as the end user, grants to the client application.

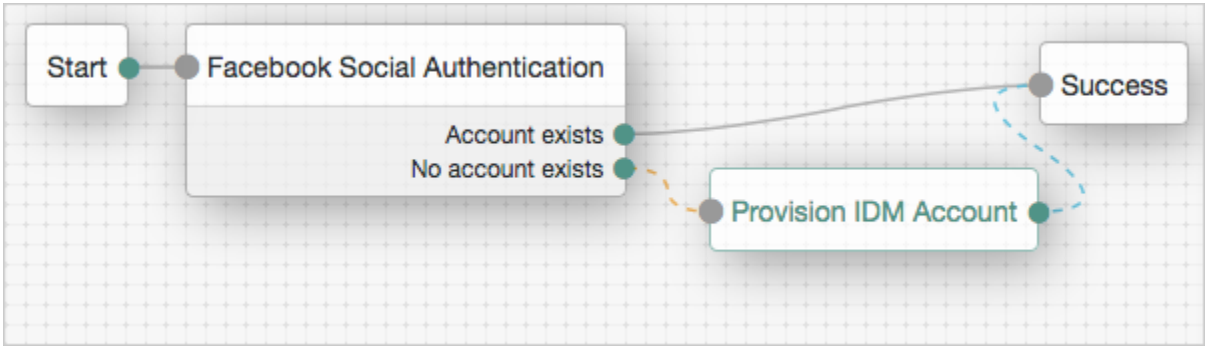
Property	Usage
Redirect URL	Specifies the URL the user is redirected to by Facebook after authenticating to continue the flow. Set this property to the URL of the AM UI. For example, `https://am.example.com:8443/am/XUI/`. **Tip** If the tree is not in the Top Level Realm, you can specify the realm in the redirect URL. Use a DNS alias for the realm, or add the realm as a query parameter, for example, `https://am.example.com:8443/am/XUI/?realm=/mySubRealm`. Learn more in Configure DNS aliases to access a realm.
Social Provider	Specifies the name of the social provider for which this node is being set up. Default: `facebook`
Auth ID Key	Specifies the attribute the social identity provider uses to identify an authenticated individual. Default: `id`
Use Basic Auth	Specifies that the client uses HTTP Basic authentication when authenticating to the social provider. Default: `true`
Account Provider	Specifies the name of the class that implements the account provider. Default: `org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider`
Account Mapper	Specifies the name of the class that implements the method of locating local accounts based on the attributes returned from Facebook. Default: `org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper`
Attribute Mapper	Specifies the list of fully qualified class names for implementations that map attributes from Facebook to AM profile attributes. Default: `org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper|uid|facebook-`

Property	Usage
Account Mapper Configuration	Specifies the attribute configuration used to map the account of the user authenticated in the Social Facebook provider to the local data store in AM. Valid values are in the form `provider-attr=local-attr`. Default: `id=uid`. **Tip** When using the `org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper` class, you can parse JSON objects in mappings using dot notation. For example, given a JSON payload of: `{ "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } }` You can create a mapper, such as `name.first_name=cn`.
Attribute Mapper Configuration	Map of Facebook user account attributes to local user profile attributes, with values in the form `provider-attr=local-attr`. Default: `name=cn`, `last_name=sn`, `id=uid`, `first_name=givenname`, `email=mail`. **Tip** When using the `org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper` class, you can parse JSON objects in mappings using dot notation. For example, given a JSON payload of: `{ "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } }` You can create a mapper, such as `name.first_name=cn`.
Save attributes in the session	When enabled, saves the attributes in the Attribute Mapper Configuration field to the AM session. Default: `true`.

Property	Usage
OAuth 2.0 Mix-Up Mitigation Enabled	Controls whether the authentication node carries out additional verification steps when it receives the authorization code from the authorization server. Specifies that the client must compare the issuer identifier of the authorization server upon registration with the issuer value returned as the `iss` response parameter. If they do not match, the client must abort the authorization process. The client must also confirm that the authorization server's response is intended for the client by comparing the client's client identifier to the value of the `client_id` response parameter. The Token Issuer property must be entered when the OAuth 2.0 Mix-Up Mitigation feature is enabled, so that the validation can succeed. The authorization code response contains an issuer value (`iss`) for the client to validate. Learn more in section 4 of OAuth 2.0 Mix-Up Mitigation Draft.
Token Issuer	Corresponds to the expected issuer identifier value in the `iss` field of the ID token. Example: `https://graph.facebook.com`

Example

The following example shows the node in context:



Social Google node

Duplicates OAuth 2.0 node, but is preconfigured to work with Google. You specify only the `Client ID` and `Client Secret`.

Note

This node and its related services, are deprecated.
You can find information on the new methods for implementing social authentication in Social authentication.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	No

Outcomes

- `Account exists`
- `No account exists`

Evaluation continues along the `Account Exists` path if an account matching the attributes retrieved from Google are found in the user data store; otherwise, evaluation continues along the `No account exists` path.

Properties

Property	Usage
Client ID *(required)*	Specifies the `client_id` parameter as provided by Google.
Client Secret *(required)*	Specifies the `client_secret` parameter as provided by Google.
Authentication Endpoint URL	Specifies the URL to the social provider's endpoint handling authentication as described in section 3.1 of The OAuth 2.0 Authorization Framework (RFC 6749). Default: `https://accounts.google.com/o/oauth2/v2/auth`
Access Token Endpoint URL	Specifies the URL to the endpoint handling access tokens as described in section 3.2 of The OAuth 2.0 Authorization Framework (RFC 6749). Default: `https://www.googleapis.com/oauth2/v4/token`
User Profile Service URL	Specifies the user profile URL that returns profile information. Default: `https://www.googleapis.com/oauth2/v3/userinfo`
OAuth Scope	Specifies a space-separated list of user profile attributes the client application requires, according to The OAuth 2.0 Authorization Framework (RFC 6749). The list depends on the permissions the resource owner, such as the end user, grants to the client application. Default: `profile email`.

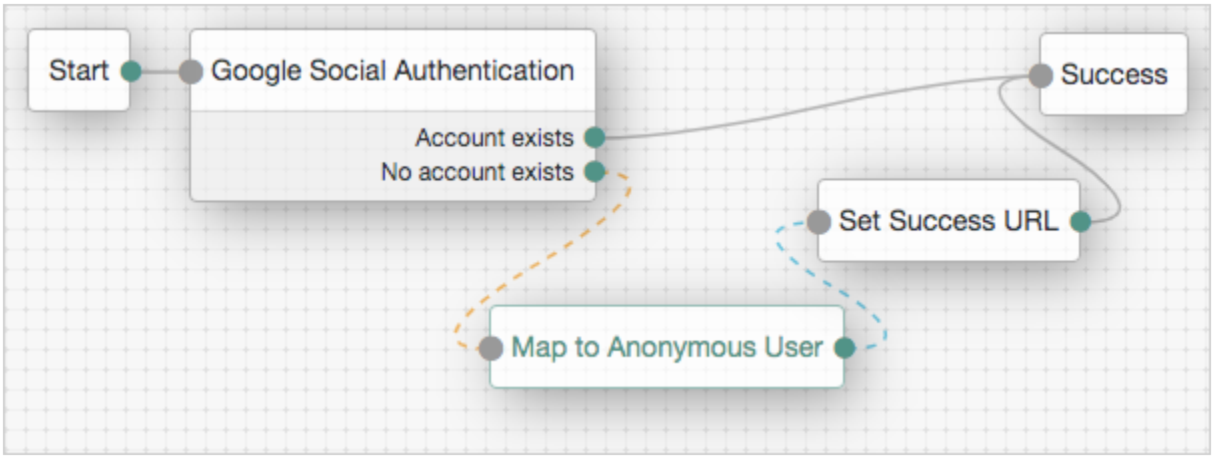
Property	Usage
Redirect URL	Specifies the URL the user is redirected to by Google after authenticating to continue the flow. Set this property to the URL of the AM UI. For example, `https://am.example.com:8443/am/XUI/`. **Tip** If the tree is not in the Top Level Realm, you can specify the realm in the redirect URL. Use a DNS alias for the realm, or add the realm as a query parameter; for example, `https://am.example.com:8443/am/XUI/?realm=/mySubRealm`. Learn more in Configure DNS aliases to access a realm.
Social Provider	Specifies the name of the social provider for which this node is being set up. Default: `google`
Auth ID Key	Specifies the attribute the social identity provider uses to identify an authenticated individual. Default: `sub`
Use Basic Auth	Specifies that the client uses HTTP Basic authentication when authenticating to Google. Default: `true`
Account Provider	Specifies the name of the class that implements the account provider. Default: `org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider`
Account Mapper	Specifies the name of the class that implements the method of locating local accounts based on the attributes returned from Google. Default: `org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper`
Attribute Mapper	Specifies the list of fully qualified class names for implementations that map attributes from Google to AM profile attributes. Default: `org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper|iplanet-am-user-alias-list|google-`

Property	Usage
Account Mapper Configuration	Specifies the attribute configuration used to map the account of the user authenticated in the Social Google provider to the local data store in AM. Valid values are in the form `provider-attr=local-attr`. Default: `sub=uid`. **Tip** When using the `org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper` class, you can parse JSON objects in mappings using dot notation. For example, given a JSON payload of: `{ "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } }` You can create a mapper, such as `name.first_name=cn`.
Attribute Mapper Configuration	Map of Google user account attributes to local user profile attributes, with values in the form `provider-attr=local-attr`. Default: `sub=uid`, `name=cn`, `given_name=givenName`, `family_name=sn`, `email=mail`. **Tip** When using the `org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper` class, you can parse JSON objects in mappings using dot notation. For example, given a JSON payload of: `{ "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } }` You can create a mapper, such as `name.first_name=cn`.
Save attributes in the session	When enabled, saves the attributes in the Attribute Mapper Configuration field to the AM session. Default: `true`.

Property	Usage
OAuth 2.0 Mix-Up Mitigation Enabled	Controls whether the authentication node carries out additional verification steps when it receives the authorization code from the authorization server. Specifies that the client must compare the issuer identifier of the authorization server upon registration with the issuer value returned as the `iss` response parameter. If they do not match, the client must abort the authorization process. The client must also confirm that the authorization server's response is intended for the client by comparing the client's client identifier to the value of the `client_id` response parameter. The Token Issuer property must be entered when the OAuth 2.0 Mix-Up Mitigation feature is enabled, so that the validation can succeed. The authorization code response contains an issuer value (`iss`) for the client to validate. Learn more in section 4 of OAuth 2.0 Mix-Up Mitigation Draft.
Token Issuer	Corresponds to the expected issuer identifier value in the `iss` field of the ID token. Example: `https://accounts.google.com`

Example

The following example shows the node in context:



Social Ignore Profile node

Specifies whether to ignore a local user profile.

If evaluation flows through this node after successful social authentication, AM issues an SSO token regardless of whether a user profile exists in the data store. AM does not check for whether a user profile is present.

Note

This node and its related services, are deprecated.
You can find information on the new methods for implementing social authentication in Social authentication.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	No

Outcomes

Single outcome path.

Properties

This node has no configurable properties.

Social Provider Handler node

The **Social Provider Handler** node attempts to authenticate a user with an identity provider they select in the Select Identity Provider node. The Social Provider Handler node collects relevant profile information from the provider, transforms the profile information into the appropriate attributes, and returns the user to the journey.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node reads the user's selected social identity provider from shared state.

Implement the Select Identity Provider node before this node to capture the social provider name.

Dependencies

- The Social Identity Provider service must be configured with the details of at least one social identity provider.
- The user must have selected a social identity provider in a previous node in the journey.

Configuration

Property	Usage
Transformation Script *(required)*	The *normalization* script of each provider maps that provider's attributes to a profile format AM can use. The *transformation* script then transforms the normalized social profile to an identity (standalone AM) or a managed object (Ping Identity Platform deployment). In standalone AM deployments, select `Normalized Profile to Identity` or a custom script that transforms the profile to an identity object. Review the sample script (normalized-profile-to-identity.js) for a list of bindings. In Ping Identity Platform deployments, select `Normalized Profile to Managed User` (default) or a custom script to transform the profile to a managed object. Review the sample script (normalized-profile-to-managed-user.js) for a list of bindings. **Important** Don't use normalization scripts (`<Identity provider>-profile-normalization.*`) for this purpose.
Username Attribute	(This property is available only in the Ping Identity Platform admin UI.) The attribute in IDM that contains the username for this object.
Client Type	The client type you're using to authenticate to the provider. Select one of the following: • `BROWSER` (default) Select this type for ForgeRock-provided user interfaces or the ForgeRock SDK for JavaScript. With this setting, the node returns the RedirectCallback. • `NATIVE` Select this type for the ForgeRock SDKs for Android or iOS. With this setting, the node returns the IdPCallback.

Outputs

- If no profile information is returned from the social provider, the journey follows the `Social auth interrupted` outcome.
- If the node retrieves profile information from the social identity provider, it transforms a normalized version of the profile and stores it in `objectAttributes` in transient state.
- In Ping Identity Platform deployments, the `aliasList` is updated and saved to `objectAttributes` in transient state to link existing users.
- The node stores the social identity subject as the `username` both directly in shared state and in its `objectAttributes`.
- The node also updates `socialOAuthData` in transient state with all existing node state, social provider data, and associated tokens.

Note

Make sure you copy required transient data to shared state because all transient data is removed if the node is followed by an interactive page later in the journey.

Outcomes

`Account exists`

Social authentication succeeded, and a matching ForgeRock account exists.

`No account exists`

Social authentication succeeded, but no matching ForgeRock account exists.

Note

In standalone AM deployments, to ensure existing users are dynamically linked, complete these additional steps:

1. Connect the `No account exists` outcome to a Scripted Decision node.
2. Write a Scripted Decision node script and use the `idRepository` binding's `get-` and `setAttribute` methods to check for an existing account and add a link by updating the account-linking attribute, `iplanet-am-user-alias-list`.
   For multiple OIDC providers, add links to the existing list. For example:

```
"iplanet-am-user-alias-list": [
    "google_IDP-123456789",
    "amazon_IDP-987654321"
],
```

3. Connect the Scripted Decision node to a Provision Dynamic Account node to update the account.

In Ping Identity Platform deployments, to ensure existing users are dynamically linked, connect the `No account exists` outcome to an Identify Existing User node followed by a Patch Object node to create the link.

`Social auth interrupted`

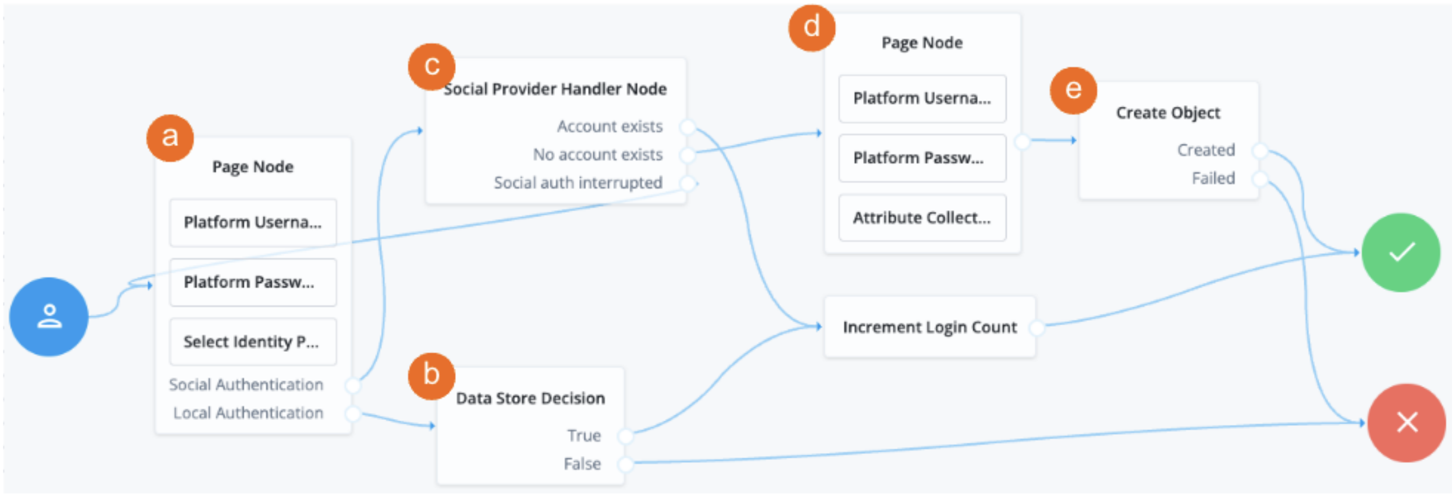
The user interrupted the social authentication journey after the node requested profile information from the social identity provider. This can happen in the following situations:

- The user clicks the **Back** button in their browser from the social identity provider's login page
- The user clicks the **Cancel** button on the social identity provider's login page
- The user re-enters the journey URL in the same browser window

  In this case, the node routes the user back to the Select Identity Provider node to select a social identity provider again.

Example

This example shows the Social Provider Handler node in a social authentication journey.



a A Page node contains the Select Identity Provider node node that prompts the user to select a social identity provider or to authenticate with a username and password.

b If the user selects local authentication, the Data Store Decision node takes care of the authentication.

c If the user selects social authentication, the Social Provider Handler node does the following:

- Routes the user to the selected social provider to authenticate there
- Retrieves the user's profile information, and transforms it into a format that AM can use
- Assesses whether the user has an existing identity in AM
- If the user has an existing identity, authenticates that identity
- If the user doesn't have an identity, routes the user to another page node
- If the user interrupts the social authentication, routes the user back to the Select Identity Provider node

d The nodes on the page node request the information required to *register* a new identity.

e The Create Object node creates the new identity in AM.

Legacy Social Provider Handler node

This legacy node is similar to the newer Social Provider Handler node. It takes a provider selection from the Select Identity Provider node and attempts to authenticate the user. The node collects relevant profile information from the provider, transforms the profile information into the appropriate attributes and returns the user to the journey.

This node remains supported in existing journeys. For new journeys, use the Social Provider Handler node instead.

Implement this node with the Select Identity Provider node to use the Social Identity Provider Service.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

`Account exists`

Social authentication succeeded, and a matching ForgeRock account exists.

`No account exists`

Social authentication succeeded, but no matching ForgeRock account exists.

Properties

Property	Usage
Transformation Script *(required)*	This script is used after the configured provider's *normalization* script has mapped the social identity provider's attributes to a profile format compatible with AM. The *transformation* script then transforms a normalized social profile to an identity (standalone AM) or a managed object (Ping Identity Platform deployment). Select `Normalized Profile to Identity` (AM standalone) or `Normalized Profile to Managed User` (Ping Identity Platform deployment), or select the custom script you've have created to transform the profile to an identity object. Find details about the script and bindings in normalized-profile-to-identity.js. Normalization scripts (`<Identity provider>-profile-normalization.*`) are not suitable for this purpose.
Username Attribute	(Ping Identity Platform deployments only.) The attribute in IDM that contains the username for this object.
Client Type	Specify the client type you are using to authenticate to the provider. Use the default, `BROWSER` , with ForgeRock-provided user interfaces or the ForgeRock SDK for JavaScript. This causes the node to return the RedirectCallback. Select `NATIVE` with the ForgeRock SDKs for Android or iOS. This causes the node to return the IdPCallback.

Write Federation Information node

Creates a persistent link between a remote IdP account and a local account in the SP, if none exists yet. If a transient link exists, it is persisted. Existing account links with different IdPs are not lost.

Use this node with the SAML2 Authentication node, and ensure that the NameID Format is `persistent`.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

Single outcome path.

Properties

This node has no configurable properties.

Find examples in SSO and SLO in integrated mode.

Identity management nodes

Accept Terms and Conditions node

The **Accept Terms and Conditions** node prompts the user to accept the currently active terms and conditions.

You set terms and conditions in the Ping Identity Platform admin UI. Learn more in Terms and conditions.

Use this node for registration, or combined with the Terms and Conditions Decision node for progressive profiling or log in.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

None.

Dependencies

This node depends on IDM for the active terms and conditions.

Configuration

This node has no configurable properties.

Outputs

The node writes a `termsAccepted` object to the shared node state. The object contains these fields:

- `acceptDate` : A timestamp string indicating when the user accepted the terms.

- `termsVersion`: A string indicating the version of the accepted terms.

Outcomes

Single outcome path; the user accepted the terms and conditions.

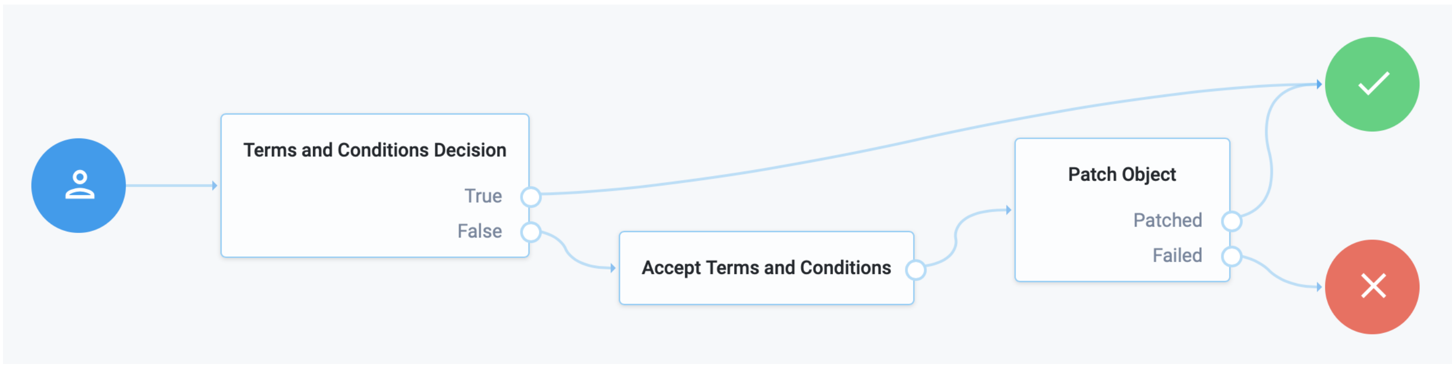
Errors

This node does not log error or warning messages of its own.

Example

For progressive profiling, include this node after a Terms and Conditions Decision node. If the user has not accepted the latest version of the terms and conditions, evaluation takes them to a page that requires them to accept the current terms and conditions.

The Patch Object node stores the acceptance response in IDM if the user accepts:



Attribute Collector node

The **Attribute Collector** node collects the values of attributes for use later in the flow; for example, to populate a new account during registration.

This node supports three types of attributes:
`string`
`boolean`
`number`

To request a value, the attribute must be present in the IDM schema of the current identity object.

The node lets you configure whether the attributes are required to continue and whether to use a policy filter of IDM to validate them.

Use the node alone or within a Page node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

For validation, this node reads the **Identity Attribute** (default: `userName`) from the shared node state. It uses the value to look up the identity object.

It prompts the user for the attributes to collect.

Dependencies

This node depends on IDM to store the user profile and perform validation.

Configuration

Property	Usage
Attributes to Collect	A list of the attributes to collect based on those in the IDM schema for the current identity object. Default: none
All Attributes Required	When enabled, all attributes collected in this node are required in order to continue. Default: false

Property	Usage
Validate Input	When enabled, validate the content against any policies specified in the IDM schema for each collected attribute. Learn more in Use policies to validate data in the IDM documentation. If you enable this property, the collected identity attributes must be *User Editable*. To make an attribute user-editable in the IDM admin UI: 1. Go to **Configure** > **Managed Objects** > ***object-name***. 2. Click the pencil (🖉) icon, then click **Show advanced options**. 3. Select the **User Editable** toggle. Learn more in [Property Configuration Properties in the IDM documentation. Default: false
Identity Attribute	The attribute used to identify the managed object in IDM. Default: `userName`

Outputs

The node writes the attributes and their values to the shared node state.

Outcomes

Single outcome path; on success, downstream nodes can read the attributes from the shared node state.

Errors

This node does not log error or warning messages of its own.

Examples

Add date and datetime fields to a journey

Note

This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.

The **Attribute Collector** node lets you add properties (attributes) that follow a date or datetime (date and time of day). The format of the date comes from the locale set in your browser.

The following table displays the differences between date and datetime:

Display format	Managed object field format	Notes
Date only	String format	The format of the date comes from the locale set in your browser. For example, if the locale is English, then the format presented to the end user is `MM-DD-YYYY`. If the local is French, the format is `DD-MM-YYYY`.
Date and time	String format (date and time of day)	The format of the date comes from the locale set in your browser. For example, if the locale is English (United States), then the format presented to the end user is `MM-DD-YYYY`. If the local is French, the format is `DD-MM-YYYY`.

Note

While the rendering of the date to the end user changes depending on the locale set in the browser, Ping Identity Platform *stores* the date value in UTC format as `YYYY-MM-DDHH:MM:SS`. For example, `2023-09-13T08:01:00Z`.

To render the date or datetime UI element to an end user with the **Attribute Collector** node, you must:

1. Specify the IDM property:
    - In the Ping Identity Platform admin UI, go to **Configure** > **Managed Objects** > *Select object*, for example, **User**.
    - Use an existing `string` property or create your own string property. Learn more in Property Configuration Properties in the IDM documentation.
2. Apply formatting and policies to the property in the IDM admin UI for `Date` or `Datetime`.

Apply formatting and policies

1. In the Advanced Identity Cloud admin UI, go to **Configure** > Managed Objects > *Select object*, for example, **User**.
2. Select the property to use.
3. To select the format of the attribute, on the **Details** tab, click the **Format** field, and select one of the following:
    - For date property — `Date`
    - For datetime property — `Datetime`
4. Click **Save**.
5. On the **Validation** tab, click **+ Add Policy**.
6. In the **Policy Id** field, enter one of the following:
    - For date property — `valid-formatted-date`
    - For datetime property — `valid-datetime`

    Learn more about applying policies to properties in Default policy reference.

7. Click **Add**.
8. In your journey, in the **Attribute Collector** node, add the property name to the **Attributes to Collect** field.

For an in-depth use case, add the date or datetime property to an **Attribute Collector** node in a registration flow. Learn more in User self-registration.

The following video shows an example of a journey collecting the datetime from an end user using the Attribute Collector node:

Your browser does not support the video tag.

Attribute Present Decision node

The **Attribute Present Decision** node checks whether an attribute is present on an object, including private attributes. There is no need to specify the value of the attribute.

Use this node during an update password flow to check whether the local account has a password, for example.

This node is similar to the Attribute Value Decision node when that node is set to use the `PRESENT` operator, except it can’t return the value of the attribute, but can work with private attributes.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node reads the **Identity Attribute** from the shared node state. If it can’t read the **Identity Attribute**, it reads the `userName` from the shared node state.

It uses the value to look up the identity object.

Dependencies

This node depends on IDM to look up the user object.

Configuration

Property	Usage
Present Attribute	The attribute whose presence you want to verify in the IDM object. This can be an otherwise private attribute, such as `password`. **Note** This field is case-sensitive and must match the IDM object attribute. For example, `givenName`, not `givenname`. Default: `password`
Identity Attribute	The attribute used to identify the managed object in IDM. Default: `userName`

Outputs

None.

Outcomes

`True`

The node found the attribute in the managed identity object.

`False`

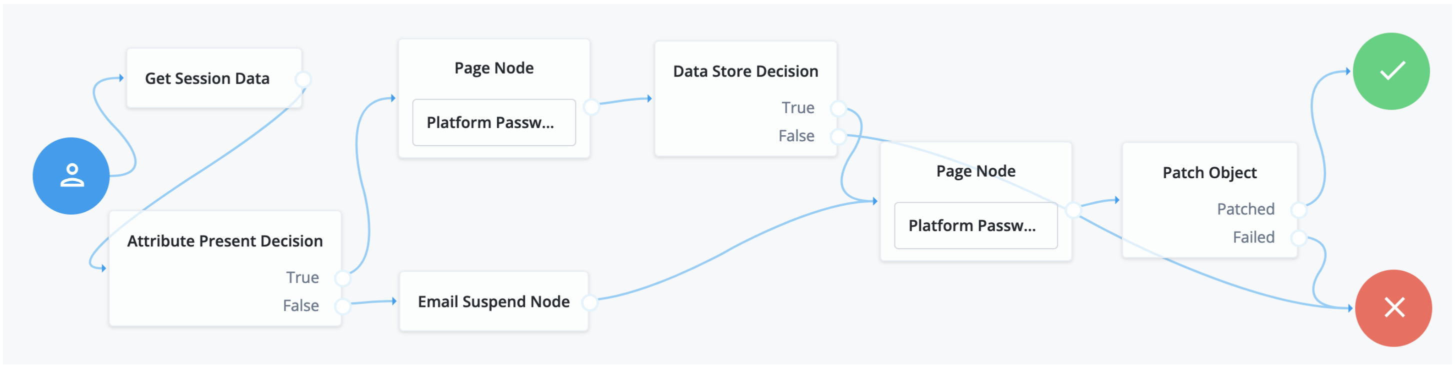
Any other case.

Errors

This node does not log error or warning messages of its own.

Example

This journey to update a password uses the **Attribute Present Decision** node to check whether the account has a password:



The user has already authenticated before beginning this journey:

- The Get Session Data node stores the `userName` from the session.
- The **Attribute Present Decision** node checks whether the user object has a password attribute.
- If so, the first Page node with the Platform Password node prompts the user for the current password.
- Otherwise, the Email Suspend node sends an email to the user and suspends the flow until the user follows the link in the message.
- The Data Store Decision node confirms the username-password credentials.
- The second Page node with the Platform Password node prompts the user for the new password.
- The Patch Object node updates the user object with the new password.

Attribute Value Decision node

Verifies that the specified attribute satisfies a specific condition.

Use this node to check whether an attribute's expected value is equal to a collected attribute value, or to validate that the specified attribute was collected.

Examples:

- To validate that a user provided the country attribute during registration, set the comparison operation to `PRESENT`, and the comparison attribute to `country`.
- To validate that the country attribute is set to the United States, set the comparison operation to `EQUALS`, the comparison attribute to `country`, and the comparison value to `United States`.

Use Attribute Present Decision node instead when you need to check for the presence of a private attribute, such as `password`.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Properties

Property	Usage
Comparison Operation	The operation to perform on the object attribute: **`PRESENT`** Checks the existence of an attribute regardless of its value. **`EQUALS`** Checks if the object's attribute value equals the configured comparison value.
Comparison Attribute	The object attribute to compare.
Comparison Value	When **Comparison Operation** is `EQUALS`, compare this value to the provided attribute value.
Identity Attribute	The attribute used to identify the managed object in IDM.

Consent Collector node

Prompts the user for consent to share their profile data.

A consent notice is listed for each identity mapping that has consent enabled. If an identity mapping is not created, or the mappings do not have privacy and consent enabled, AM does not show a consent message to the user.

This node is primarily used in progressive profile and registration flows.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Properties

Property	Usage
All Mappings Required	If enabled, all mappings listed by this node require consent in order to move forward.
Privacy & Consent Message	Localized message providing the privacy and consent notice. The key is the language, such as `en` or `fr`, and the value is the message to display.

Create Object node

The **Create Object node** lets you create a new object in IDM based on information collected during authentication, such as user registration.

Any managed object attributes that are marked as required in IDM must be collected during authentication in order to create the new object.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node requires the managed object attributes marked as required

Configuration

Property	Usage
Identity Resource	The type of managed identity resource object that this node creates. It must match the identity resource type for the current flow. **Tip** To check for the available managed identity resource types, go to the IDM admin UI, and open the **Manage** drop-down list in the upper right corner of the screen. Identity managed object types are preceded by the 👤 icon. Default: `managed/user`

Outputs

This node doesn't change the shared state.

Outcomes

This node has the following outcomes:

- Created
- Failed

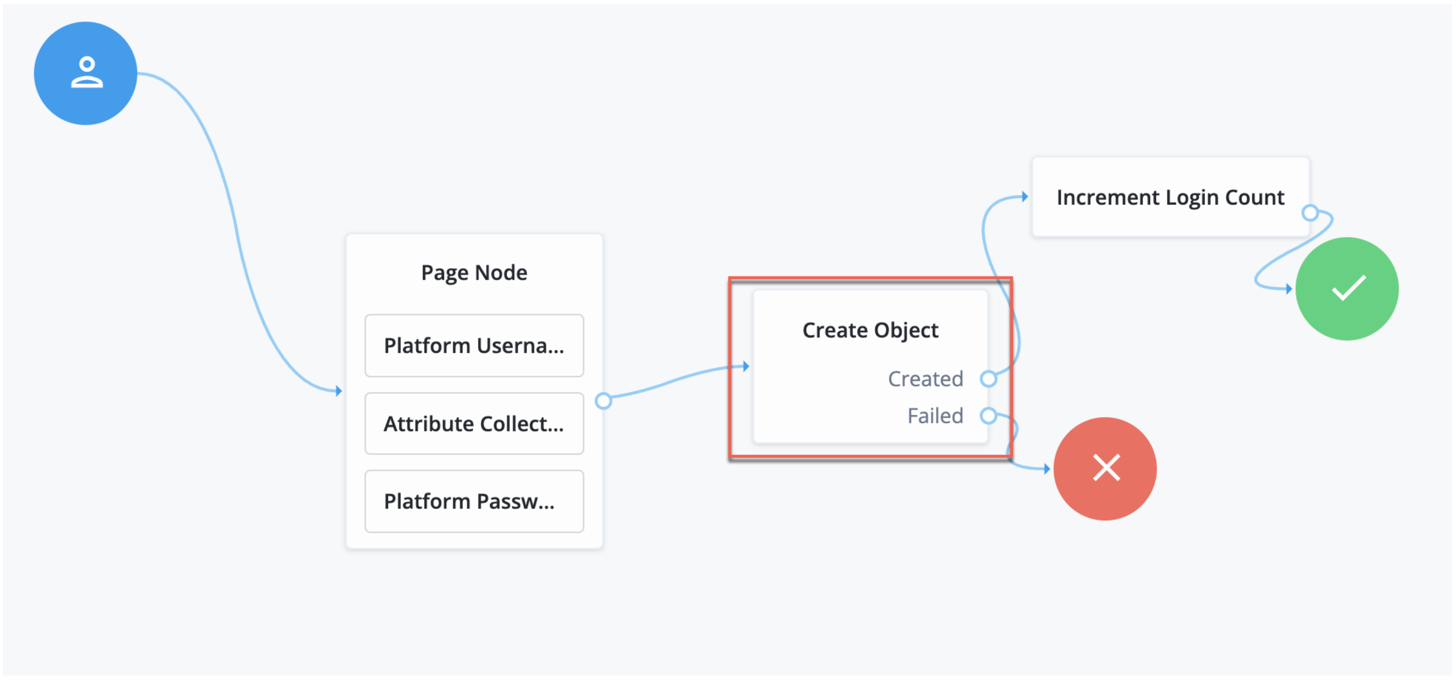
Errors

This node can log the following warning messages:

Message	Notes
Failed to create object	The preceding nodes don't provide all the fields required to create the object.
Failed to retrieve object's schema	The node failed to get the list of required attributes from the **Identity Resource** schema.

Example

The following example uses this node with the Increment Login Count node to create a new user object.



- The Page node includes various nodes that collect attributes and store them in the shared node state.
- The Create Object node uses these attributes to create the new user.
- The Increment Login Count node resets the retry count on successful authentication of the new user.

Create Password node

Lets users create a password when provisioning an account.

Note

This node and its related services, are deprecated.
You can find information on the new methods for implementing social authentication in Social authentication.

Social identity providers do not provide a user's password. Use this node to provide a password to complete the user's credentials before provisioning an account.

Important

The flow must provision an account after prompting the user for a password, for example, by using the Provision Dynamic Account node. If no account is provisioned, the flow does not save the password.
Do not place any nodes that request additional input from the user between this node and the provisioning node; otherwise, the password is lost.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	No

Outcomes

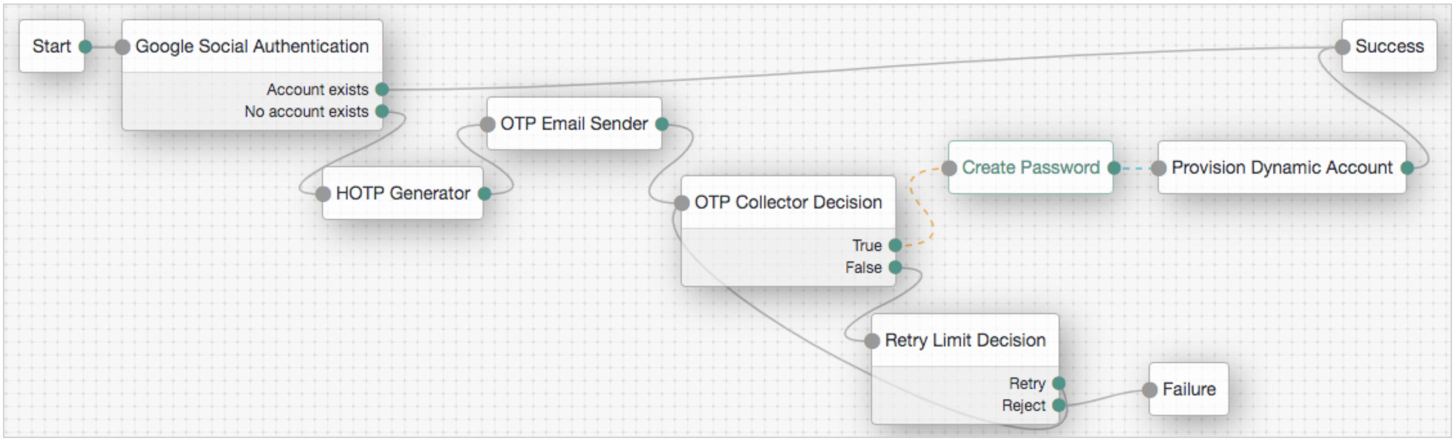
Single outcome path.

Properties

Property	Usage
minPasswordLength	Specifies the minimum number of characters the password must contain.

Example

The following example lets users who have performed social authentication using Google provide a password and provision an account when they don't have one. They must enter a one-time passcode to verify they are the owner of the Google account.



Display Username node

Fetches a username based on a different identifying attribute, such as an email address, then displays it.

To email the username to the user instead, use the Identify Existing User node combined with a Email Suspend node or Email Template node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Properties

Property	Usage
User Name	The attribute used to identify the username in an IDM object.
Identity Attribute	The attribute used to identify the managed object in IDM. When this node serves to recover a username, the identity attribute should be some other attribute that is unique to a user object, such as the email address. The node raises an exception when more than one value exists for this attribute. Make sure the value of whatever attribute you select is unique for each user.

Identify Existing User node

The **Identify Existing User** node verifies if a user exists based on an identifying attribute, such as an email address, then makes the value of a specified attribute available in the shared node state.

Use this node in a forgotten password flow to fetch a username to email to the user. To display the username on the screen, use the Display Username node instead.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes

Product	Available?
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node reads the **Identity Attribute** (default: `mail`) from the shared node state.

If the **Identity Attribute** is not available, it reads the `userName` from the shared node state.

Dependencies

This node depends on IDM to store the user profile.

Configuration

Property	Usage
Identifier	The attribute to collect from a managed identity object. Default: `userName`
Identity Attribute	The attribute used to identify the managed object in IDM. When this node serves to recover a username, the identity attribute should be some other attribute that is unique to a user object, such as the email address. Default: `mail`

Outputs

The node writes the **Identifier** and the user account `_id` to the shared node state.

If the **Identifier** differs from `userName`, this node also writes the `userName` to the shared node state.

Outcomes

`True`

The node successfully identified the user and updated the shared node state.

`False`

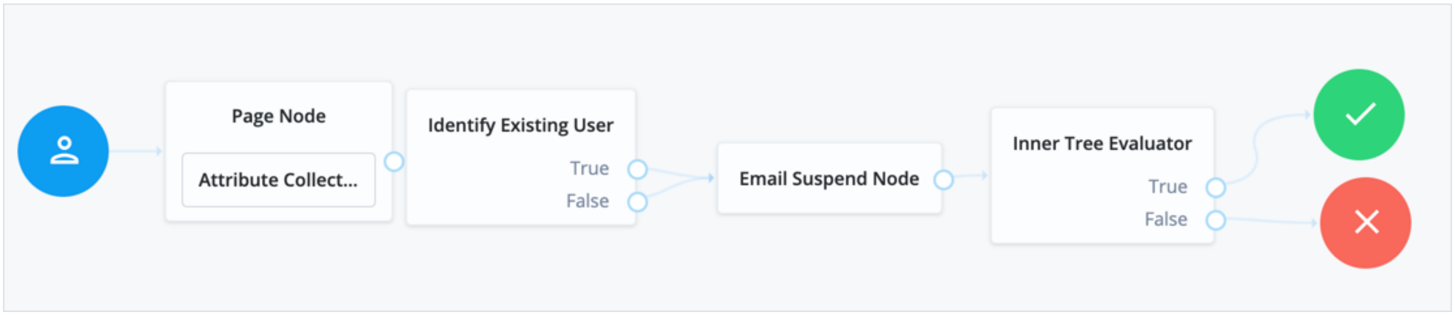
Any other case.

Errors

This node does not log error or warning messages of its own.

Example

The following example shows a flow to reset a forgotten password:



- The user enters their email in the Attribute Collector node of the Page node.
- The **Identify Existing User** node uses the email address to look up the username of the user's account. If it finds the user account, it adds the username to the shared node state.
- The Email Suspend node emails the user and suspends authentication.
- Once authentication resumes, the Inner Tree Evaluator node sends the user to a different flow to reset their password.

KBA Decision node

Checks whether the user account has the required minimum number of KBA questions.

To set the number of KBA questions, edit **Configure** > **Security Questions** > **Questions** > **Number** in the IDM admin UI.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) ⓘ **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes

Product	Available?
Ping Identity Platform (self-managed)	Yes

Outcomes

- `True`
- `False`

Evaluation continues along the `True` path if the user profile holds at least the minimum number of KBA questions; otherwise, evaluation continues along the `False` path.

Properties

Property	Usage
Identity Attribute	The attribute used to identify the managed object in IDM.

KBA Definition node

The **KBA Definition** node collects knowledge-based authentication (KBA) questions and answers.

Use this node when creating or updating a user with KBA enabled. Learn more in Security questions.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

None. This node doesn't require any attributes from the shared node state.

Dependencies

This node depends on IDM for the KBA configuration.

Configuration

Property	Usage
Purpose Message	A localized message describing the purpose of the data requested from the user. Default: none
Allow User-Defined Questions	When enabled, users can create their own KBA questions. Disable this setting to restrict users to select from predefined questions only. Default: Enabled
Questions	Create or modify custom localized questions that the user can choose from when defining security questions. To add a localized security question: 1. Click **+** to open the **Add a Security Question** form. 2. Select from the list of existing locales or add a new locale, type a question into the text field, and click **Done**. 3. Repeat to add further questions, and click **Save** when complete. To edit an existing security question, click the edit icon ✎, make your changes, and click **Save**. Default: `What's your favorite color?` (locale: `en`)

Outputs

The node writes the KBA questions and answers in the transient shared node state.

Outcomes

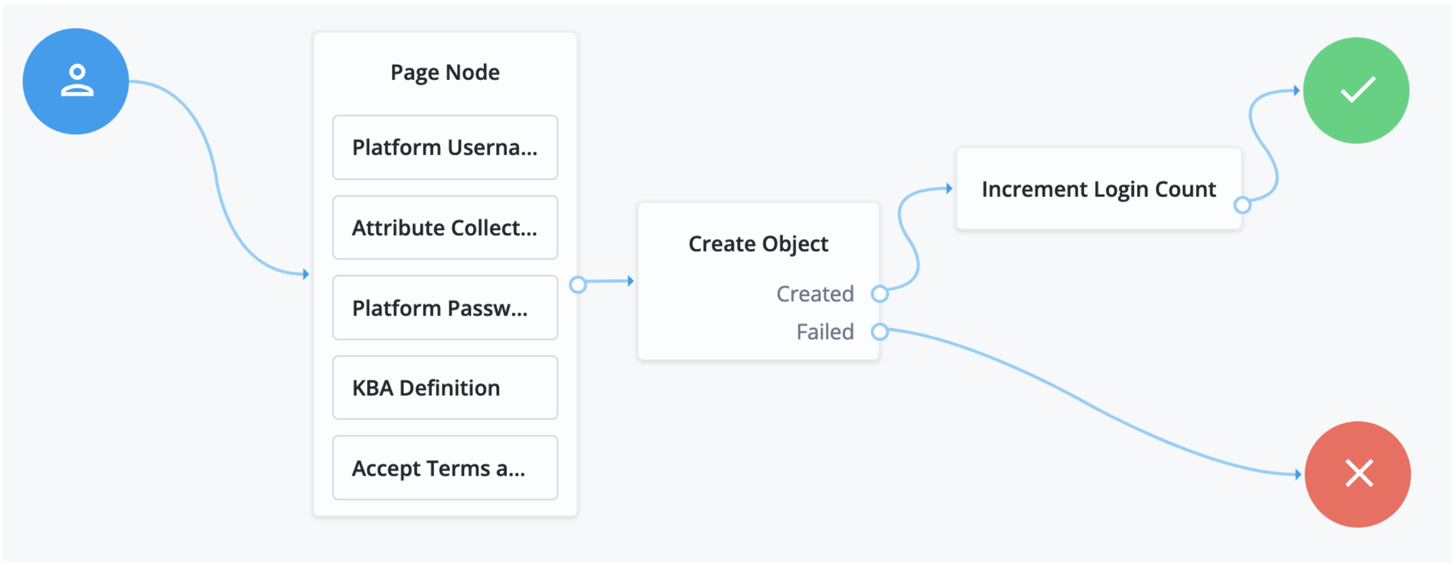
Single outcome path; on success, the transient state holds the questions and answers.

Errors

This node logs a `Failed to retrieve kba configuration` warning message when it can't read the configuration.

Example

The following registration journey prompts for questions and answers when creating an account:



- The Page node collects registration information:
    - The Platform Username node prompts for and collects a username for the new account.
    - The Attribute Collector node prompts for a given name, a surname, an email address, and profile preferences.
    - The Platform Password node prompts for and collects a password.
    - The **KBA Definition** node collects questions and answers.
    - The Accept Terms and Conditions node prompts the user to accept the active terms and conditions.
- The Create Object node stores the collected information in the new account object.
- The Increment Login Count node updates the number of successful authentications.

KBA Verification node

Presents KBA questions to the user, collects answers to those questions, and verifies the input against the user's stored answers.

Use this node for additional authentication when resetting a forgotten password or username.

To set the number of KBA questions, edit **Configure** > **Security Questions** > **Questions** > **Number** in the IDM admin UI.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes

Product	Available?
PingAM (self-managed) ⓘ **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Properties

Property	Usage
KBA Attribute	The managed object attribute in which KBA questions and answers are stored.
Identity Attribute	The attribute used to identify the managed object in IDM.

Pass-through Authentication node

Authenticates an identity through a connector to a third-party service.

This lets you migrate user profiles without forcing users to reset their passwords, or retain a third-party service indefinitely as the canonical store for authentication credentials.

Before you use the node:

- Configure the connector to the third-party service.

  Find details in the OpenICF documentation.

- If you plan to collect credentials in the identity repository for users, synchronize accounts from the third-party service.

  Learn more in Synchronization in the IDM documentation.

This node requires the `username` and `password` properties in the incoming node state. Implement the following nodes earlier in the journey:

- Standalone PingAM deployment
  - Username Collector node
  - Password Collector node
- Ping Identity Platform deployment:
  - Platform Username node
  - Platform Password node

Pass the credentials to this node to authenticate the identity against the service.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Connectors that support pass-through authentication

The following connectors support pass-through authentication using the `AuthenticateOp` interface by default:

- LDAP connector
- CSV file connector
- Database Table connector
- Microsoft Graph API Java connector
- Scripted SQL connector

Note

All Scripted Groovy-based connectors are capable of pass-through authentication if the `AuthenticateScript.groovy` script is implemented, but the only default implementation is the ScriptedSQL connector. For more information, refer to Authenticate script and Authenticate operation.

Outcomes

- `Authenticated`
- `Missing Input`
- `Failed`

Properties

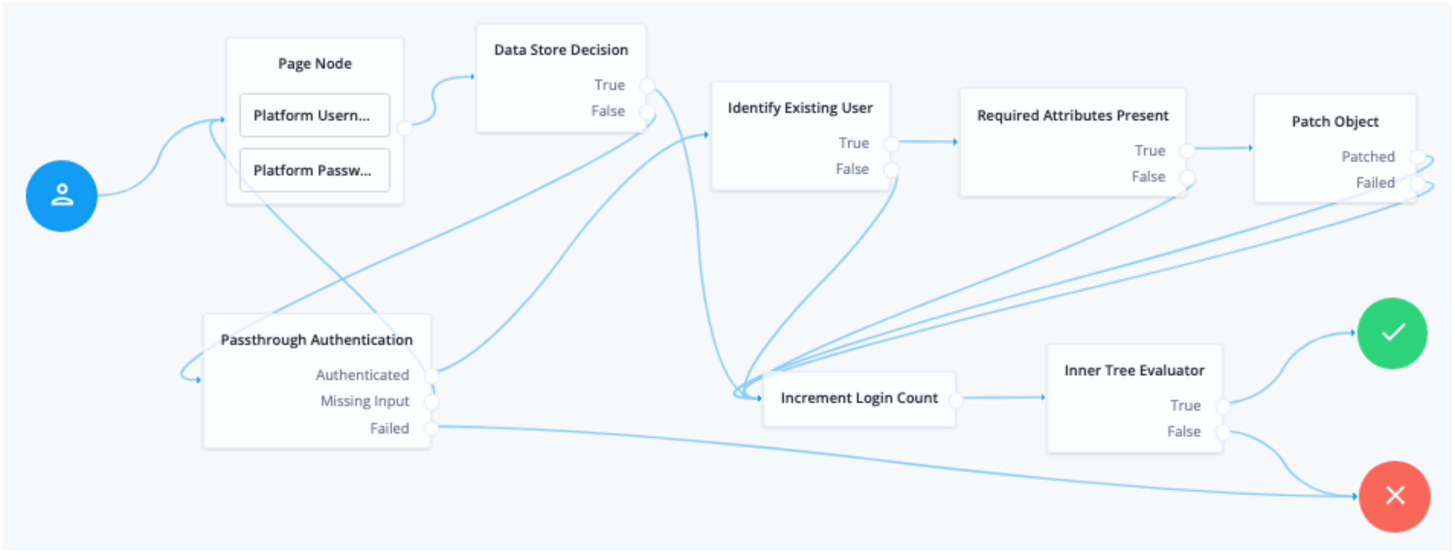
Property	Usage
System Endpoint	Required. Name of the connector to the third-party service that performs authentication.
Object Type	The OpenICF object type for the object being authenticated. Default: `account`
Identity Attribute	The username attribute for authentication. Default: `userName`
Password Attribute	The password attribute for authentication. Default: `password`

Example

The following example requires a Ping Identity Platform deployment.

Before trying this example, synchronize accounts from the third-party service. The example shows a login flow that tries pass-through authentication when local authentication fails, and stores the user password when authentication with the third-party service succeeds.

In this example, the user enters their credentials with the Platform Username node and Platform Password node. The Data Store Decision node authenticates against the platform directory service. On failure, authentication passes through to the third-party service. If authentication with the third-party service is successful, the Identify Existing User node and Required Attributes Present node check for a valid user profile. The Patch Object node updates the user's profile with the successful password:



List of node connections

Source node	Outcome path	Target node
Page Node containing: • Platform Username • Platform Password	→	Data Store Decision
Data Store Decision	True	Increment Login Count
	False	Pass-through Authentication
Pass-through Authentication	Authenticated	Identify Existing User
	Missing Input	Page Node
	Failed	Failure
Identify Existing User	True	Required Attributes Present
	False	Increment Login Count
Required Attributes Present	True	Patch Object
	False	Increment Login Count
Patch Object	Patched	Increment Login Count
	Failed	Increment Login Count
Increment Login Count	→	Inner Tree Evaluator
Inner Tree Evaluator	True	Success
	False	Failure

Patch Object node

The **Patch Object** node updates the attributes of an existing managed identity object.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes

Product	Available?
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node reads the **Identity Attribute** and the managed object fields to patch from the shared node state. If it can't read the **Identity Attribute**, it reads the `userName` from the shared node state.

Dependencies

This node depends on IDM to find and patch the managed object.

Configuration

Property	Usage
Patch as Object	If enabled, update the object as its subject—for example, update a managed user object as the user; otherwise, update the object as the client application. Enable this property to patch fields of the current, authenticated user's account the client application can't update. Default: false
Ignored Fields	Omit the specified shared state fields from the patch. If no fields are specified, the node attempts to update all the shared state fields as part of the patch. Default: none
Identity Resource	The type of managed identity resource object this node patches. This must match the identity resource type for the current flow. **Tip** To check for the available managed identity resource types, go to the IDM admin UI, and open the **Manage** drop-down list in the upper right corner of the screen. Identity managed object types are preceded by the 👤 icon. Default: `managed/user`
Identity Attribute	The attribute used to identify the managed object in IDM. Default: `userName`

Outputs

None.

Outcomes

`Patched`

The node updated the managed object.

`Failed`

Any other case.

Errors

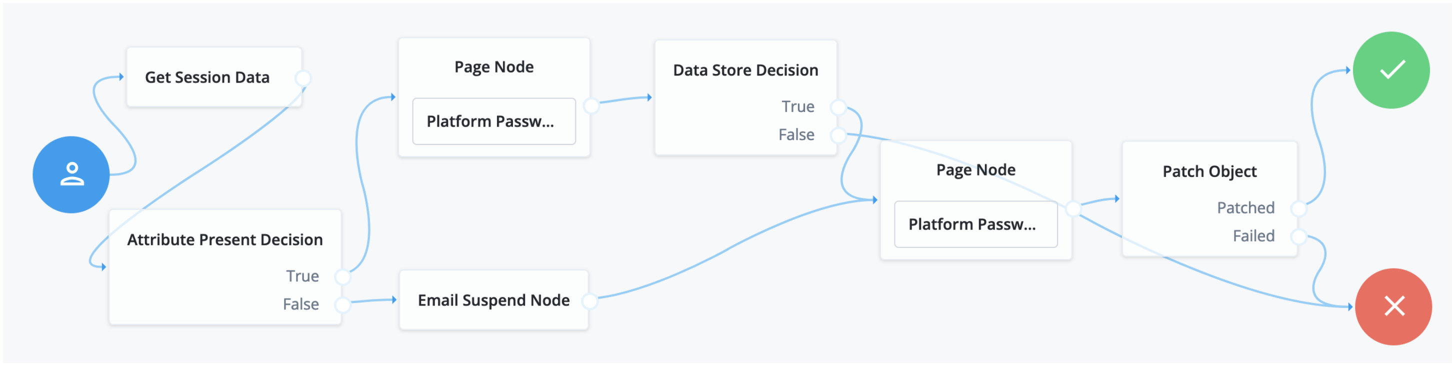
This node logs the following warning messages when an update fails:

- `Failed to create object`
- `Failed to patch object`

Review the logs for additional messages describing the problem.

Example

This journey uses the **Patch Object** node to update a user's password:



The user has already authenticated before beginning this journey:

- The Get Session Data node stores the `userName` from the session.
- The Attribute Present Decision node checks whether the user object has a password attribute.
- If so, the first Page node with the Platform Password node prompts the user for the current password.
- Otherwise, the Email Suspend node sends an email to the user and suspends the flow until the user follows the link in the message.
- The Data Store Decision node confirms the username-password credentials.

- The second Page node with the Platform Password node prompts the user for the new password.
- The **Patch Object** node updates the user object with the new password.

Platform Password node

The **Platform Password** node prompts the user to enter their password and stores it in a configurable property of the shared node state.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node uses the `_id` of the object for policy evaluation.

For existing users, the user's `_id` must be in the shared state to evaluate user-specific policies, such as password history, cannot-contain-others, and so on. No `_id` is available for new users.

Dependencies

If this node's **Validate Password** setting is enabled, the node relies on IDM for password policies.

Configuration

Property	Usage
Validate Password	When enabled, this node uses the password policies in IDM to validate the user's input. It returns any policy failures as errors. For example, if you submitted an invalid password on registration, the response from this node would include a list of failed policies: `{ "name": "failedPolicies", "value": [ "{ \"params\": { \"minLength\": 8 }, \"policyRequirement\": \"MIN_LENGTH\" }", "{ \"params\": { \"numCaps\": 1 }, \"policyRequirement\": \"AT_LEAST_X_CAPITAL_LETTERS\" }", "{ \"params\": { \"numNums\": 1 }, \"policyRequirement\": \"AT_LEAST_X_NUMBERS\" }" ] }` Default: disabled
Password Attribute	The attribute used to store a password in the managed identity object. Default: `password`
Confirm Password	Enable this option to require the user to enter the password identically in a second field. **Note** This property only displays when the node is placed within a Page node in a Ping Identity Platform deployment. Default: disabled
Checkmark Policy Display	Enable this option to show a checkmark instead of faded bullet points on successful password validation. **Note** This property only displays when the node is placed within a Page node in a Ping Identity Platform deployment. Default: disabled

```
{
    "name": "failedPolicies",
    "value": [
        "{ \"params\": { \"minLength\": 8 }, \"policyRequirement\":
\"MIN_LENGTH\" }",
        "{ \"params\": { \"numCaps\": 1 }, \"policyRequirement\":
\"AT_LEAST_X_CAPITAL_LETTERS\" }",
        "{ \"params\": { \"numNums\": 1 }, \"policyRequirement\":
\"AT_LEAST_X_NUMBERS\" }"
      ]
}
```

Outputs

On success, this node updates the **Password Attribute** property in the shared node state with the password.

The captured password is transient, persisting only until the authentication flow reaches the next node requiring user interaction. It may be persisted to the secure state if required later in the journey.

Outcomes

Single outcome path.

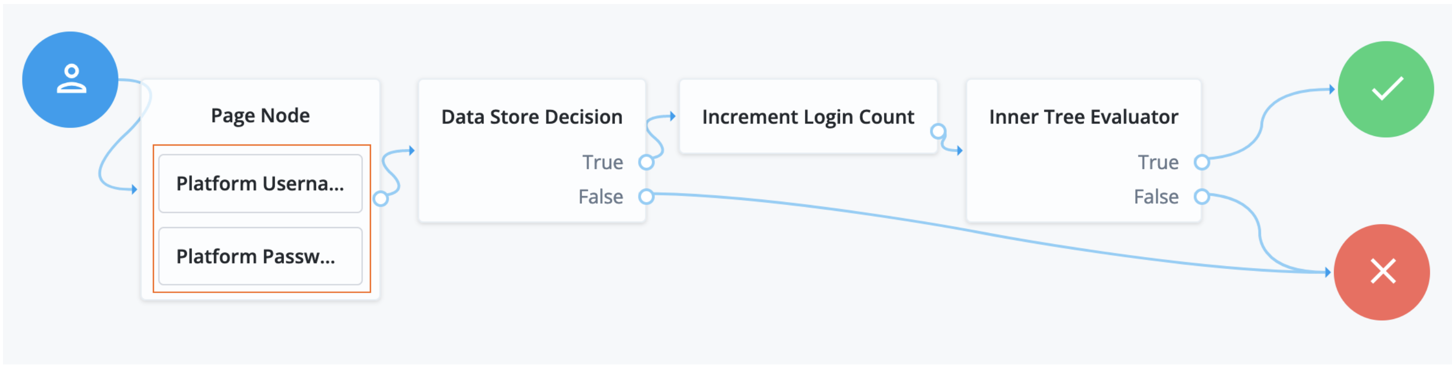
Errors

This node does not log error or warning messages of its own.

If it fails to get the result from IDM for a validation request, this node throws an exception with a `Communication failure` message.

Example

The following journey uses a Page node containing the Platform Username node and Platform Password node to collect the username and password and set their values in the shared node state:



- The Page node presents a page with input fields to prompt for the username and password.
  - The Platform Username node collects and injects the `userName` into the shared node state.
  - The Platform Password node collects and injects the `password` into the shared node state.
- The Data Store Decision node uses the username and password to determine whether authentication is successful.
- The Increment Login Count node updates the login count on successful authentication.
- The Inner Tree Evaluator node invokes a nested journey for progressive profiling.

Platform Username node

The **Platform Username** node prompts the user to enter their username and stores it in a configurable property of the shared node state.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

None.

Dependencies

If this node's **Validate Username** setting is enabled, the node relies on IDM for username policies.

Configuration

Property	Usage
Validate Username	When enabled, this node uses the username policies in IDM to validate the user's input. It returns any policy failures as errors. **Important** Only enable this field if you're using this node as part of a *registration* journey. Don't enable this field in an *authentication* journey because the validation includes verifying that the provided username doesn't exist in the identity store. Default: disabled
Username Attribute	The attribute used to store a username in the managed identity object. Default: `userName`

Outputs

On success, this node updates the **Username Attribute** property in the shared node state with the username.

Outcomes

Single outcome path.

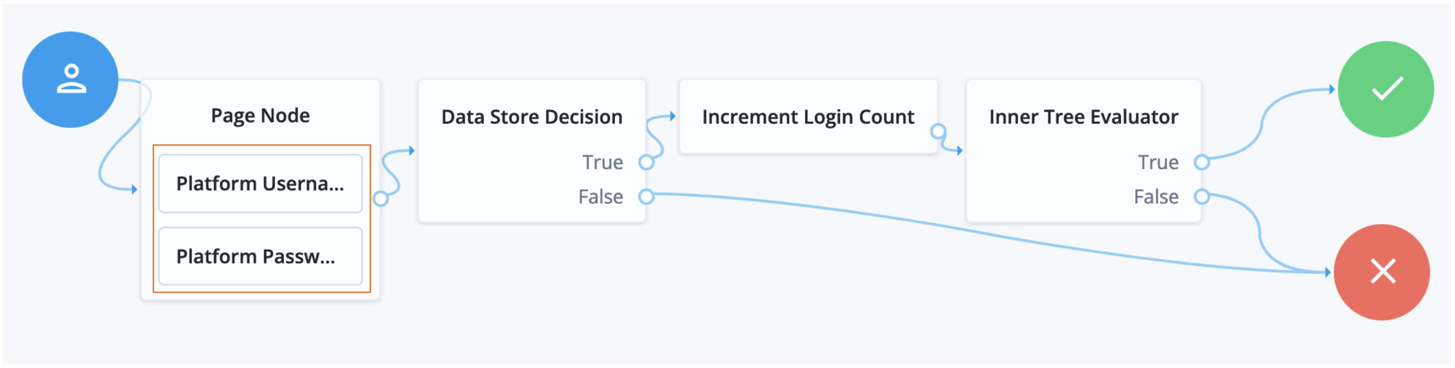
Errors

This node does not log error or warning messages of its own.

If it fails to get the result from IDM for a validation request, this node throws an exception with a `Communication failure` message.

Example

The following journey uses a Page node containing the Platform Username node and Platform Password node to collect the username and password and set their values in the shared node state:



- The Page node presents a page with input fields to prompt for the username and password.
  - The Platform Username node collects and injects the `userName` into the shared node state.
  - The Platform Password node collects and injects the `password` into the shared node state.
- The Data Store Decision node uses the username and password to determine whether authentication is successful.
- The Increment Login Count node updates the login count on successful authentication.
- The Inner Tree Evaluator node invokes a nested journey for progressive profiling.

Profile Completeness Decision node

Use progressive profile flows to check how much of a user’s profile has been completed, where the completeness of a profile is expressed as a percentage of user-viewable, and user-editable fields that are not `null`.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `True`
- `False`

Properties

Property	Usage
Profile Completeness Threshold	Percentage of user-viewable and user-editable fields in a profile that must be filled for the node to pass. Express this as a number between 0 and 100.
Identity Attribute	The attribute used to identify the managed object in IDM.

Query Filter Decision node

The **Query Filter Decision** node checks if the contents of a user's profile match the specified query filter.

Use this node to check whether an attribute of the user profile matches a specific pattern. For instance, use this in progressive profile flows to check if marketing preferences are set on a user's profile.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes

Product	Available?
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node reads the **Identity Attribute** from the shared node state. If it can't read the **Identity Attribute**, it reads the `userName`.

It uses the value to look up the identity object.

Dependencies

This node depends on IDM to look up the user object.

Configuration

Property	Usage
Query Filter	A query filter used to check the contents of an object. Find details on constructing effective query filters in Construct queries in the IDM documentation. Default: none
Identity Attribute	The attribute used to identify the managed object in IDM. Default: `userName`

Outputs

None.

Outcomes

`True`

The node user profile matched the query.

`False`

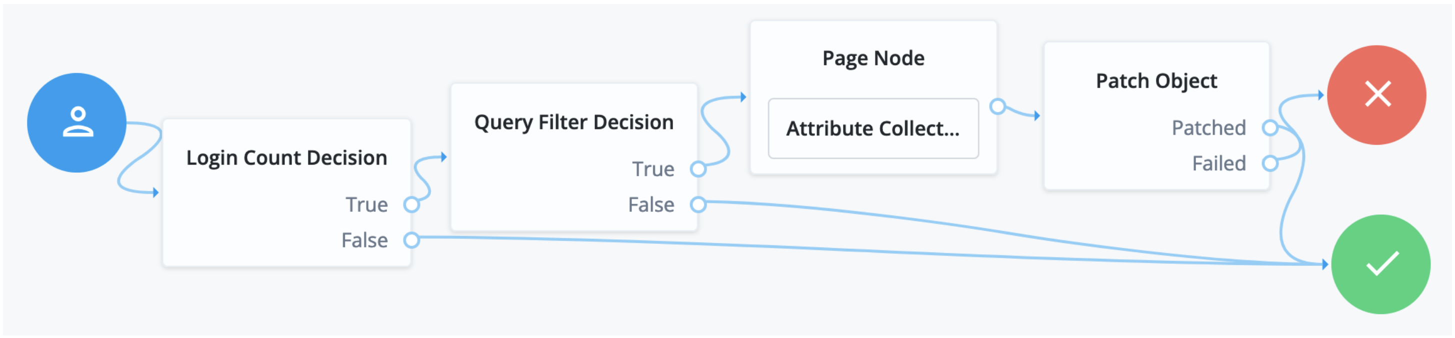
Any other case.

Errors

This node doesn't log error or warning messages of its own.

Example

Other journeys invoke the following progressive profile journey to capture missing profile attributes:



- The Login Count Decision node triggers the rest of the journey depending on the login count and its settings.
- The **Query Filter Decision** node determines whether managed object profile fields are missing.
- The Attribute Collector node in the Page node requests additional input for the profile.
- The Patch Object node stores the additional input in the managed object profile.

Required Attributes Present node

Checks the specified identity resource in IDM, by default, `managed/user`, and determines if all attributes required to create the specified object exist within the shared node state.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) ⓘ **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `True`
- `False`

Properties

Property	Usage
Identity Resource	The type of managed identity resource object this node creates. It must match the identity resource type for the current flow. **Tip** To check for the available managed identity resource types, go to the IDM admin UI, and open the **Manage** drop-down list in the upper right corner of the screen. Identity managed object types are preceded by the icon.

Select Identity Provider node

Presents the user with a list of configured, enabled, social identity providers to use for authentication.

Use this node with the Social Provider Handler node to use the Social Identity Provider Service.

This node can be configured to only show identity providers the user has already associated with their account, such as in account claiming flows, where a user wishes to associate a new social identity provider with an account that is being authenticated with social authentication.

The node has two possible outputs: social authentication and local authentication. Local authentication can be turned off by disabling **Include local authentication**.

In cases such as account claiming, where the user has already authenticated once and is associating a new identity provider, the node only displays a local sign in option if it detects that the user's account has a `password` attribute.

This node returns the SelectIdPCallback when more than one social identity provider is enabled, or a single provider is enabled as well as the **Local Authentication** option, It then requires a choice from the user. If no choice from the user is required, authentication proceeds to the next node in the flow.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes

Product	Available?
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `Social Authentication`
- `Local Authentication`

To turn off local authentication, disable **Include local authentication**.

Properties

Property	Usage
Include local authentication	Whether local authentication is included as a method for authenticating.
Offer only existing providers	Ping Identity Platform deployments only. Enable this when the social identity provider choices offered should be limited to those already associated with a user object. Use this when a user is authenticating using a new social identity provider, and an account associated with that user already exists (also known as "account claiming").
Password attribute	Ping Identity Platform deployments only. The attribute in the user object that stores a user's password for use during local authentication.
Identity Attribute	Ping Identity Platform deployments only. The attribute used to identify an existing user. Required to support the offer of only existing providers.
Filter Enabled Providers	By default, the node displays all identity providers marked as **Enabled** in the Social Identity Provider Service as a selectable option. Specify the name of one of more providers to filter the list. **Tip** View the names of your configured social identity providers in AM admin UI under **Realms** > ***Realm name*** > **Services** > **Social Identity Provider Service** > **Secondary Configurations**. If this field is not empty, providers must be in the list and must be enabled in the Social Identity Provider service to appear. If left blank, the node displays all enabled providers.

Terms and Conditions Decision node

Verifies the user has accepted the active set of terms and conditions.

You set up terms and conditions in the Ping Identity Platform admin UI. Learn more in Terms and conditions.

Use this node to verify the user has accepted terms and conditions before proceeding, for example, during login or progressive profile data collection.

You can use this node with the Accept Terms and Conditions node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `True`
- `False`

Properties

Property	Usage
Identity Attribute	The attribute used to identify the managed object in IDM.

Time Since Decision node

Checks if a specified amount of time has passed since the user was registered.

For example, to prompt users to review your terms and conditions after the account is a week old, set the **Elapsed Time** property to `1 week`. After that time has elapsed, the next time the user logs in, they are prompted to review your terms and conditions.

Use this node for progressive profile completion.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `True`
- `False`

Properties

Property	Usage
Elapsed Time	The amount of time since the user was created, in minutes, that needs to elapse before this node is triggered. This property also supports specifying basic time units. For example, when setting the property to `10080` minutes, writing `7 days` or `1 week` also works.
Identity Attribute	The attribute used to identify the managed object in IDM.

Utility nodes

Agent Data Store Decision node

The **Agent Data Store Decision** node authenticates the agent using the data store for agent profiles and sets its authentication identifier if successful.

Note

This node only authenticates agents, such as PingGateway and AM Java and web agents.
Use the Data Store Decision node for other identities.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node requires the `username` and `password` properties in the incoming node state.

Obtain the agent credentials from the user or with a Zero Page Login Collector node.

Dependencies

This node depends on the underlying data store for agent profiles.

Configuration

This node has no configurable properties.

Outputs

This node copies the shared and transient states into the outgoing node state.

Outcomes

`True`

The credentials match those found in the data store for agent profiles.

`False`

The credentials do *not* match those found in the data store for agent profiles.

Errors

This node can log the following warnings:

`Exception in data store decision node`

The node couldn't connect to the data store, or another error occurred.

`invalid password error`

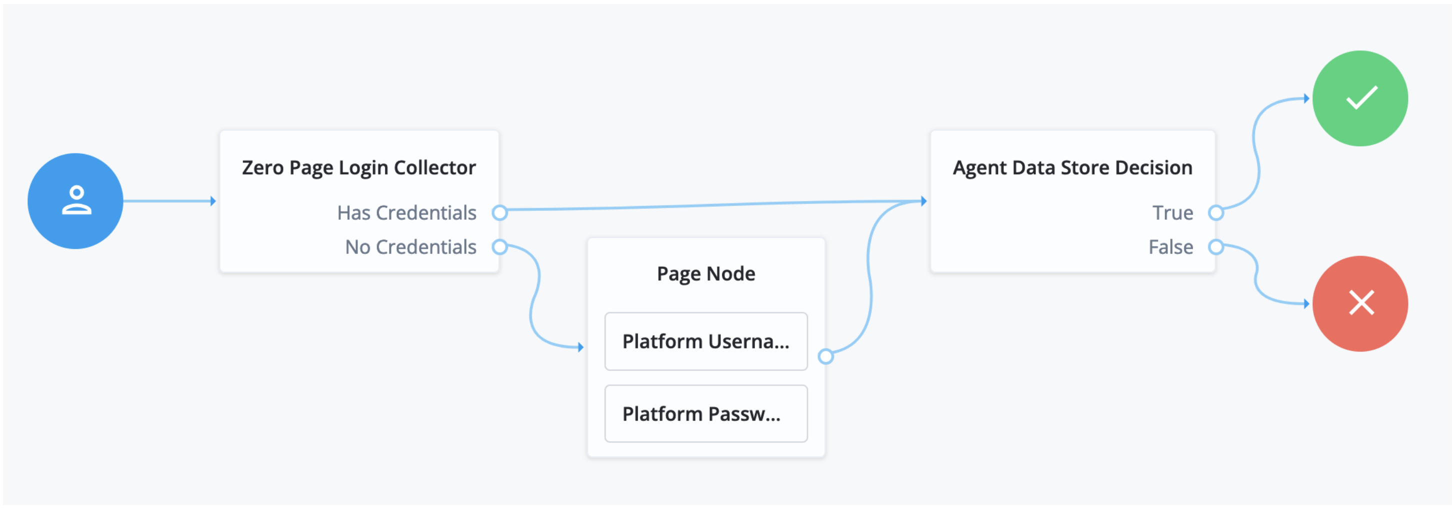
The password doesn't match.

`invalid username error`

The username doesn't match any profiles found in the data store.

Example

The following example uses this node to authenticate an agent with the credentials provided:



- The Zero Page Login Collector node collects the username and password from HTTP headers if provided.
- The Page node collects the username and password interactively from the user.
- The **Agent Data Store Decision** node verifies the agent credentials match those in the data store.

Anonymous Session Upgrade node

Upgrades an anonymous session to a non-anonymous session.

Use this as the first node in the flow.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

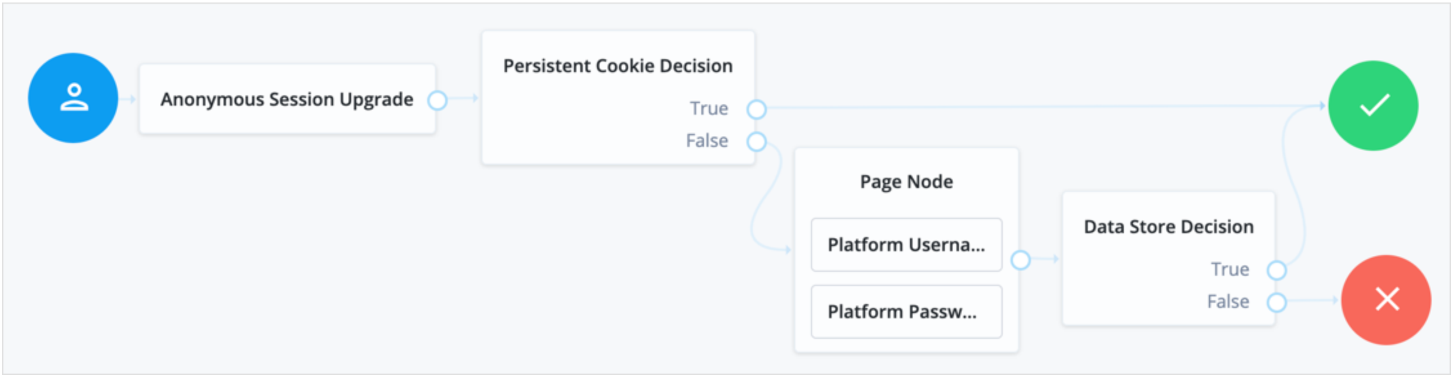
Single outcome path.

Properties

This node has no configurable properties.

Example

After using the Anonymous User Mapping node to access AM as an anonymous user, this node lets users upgrade their session to a non-anonymous one:



Anonymous User Mapping node

Lets users log in to an application or website without providing credentials, by assuming the identity of a specified existing user account. The default user for this purpose is named `anonymous`.

Take care to limit access for such users. For example, grant anonymous users access to public downloads on your site.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

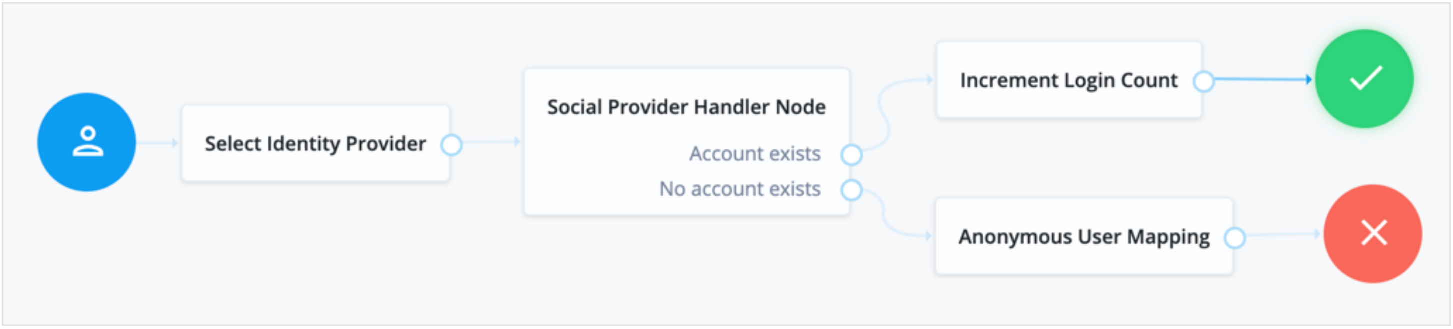
Single outcome path.

Properties

Property	Usage
Anonymous User Name	Specifies the username of an account that represents anonymous users. This user must already exist in the realm, and its user status must be `active`.

Example

The following example uses this node to grant access as an anonymous user to users who have performed social authentication access and do not have an existing profile:



Choice Collector node

Define two or more options to present to the user when authenticating.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- *Choice 1*

  …

- *Choice n*

Properties

Property	Usage
Choices	Enter two or more choice strings to display to the user. To remove a choice, select its **Delete** icon ✕. To delete all choices, select the **Clear all** button in the **Choices** field.
Default Choice *(required)*	Enter the value of the choice to be selected by default. **Important** If you do not specify a default choice, the first choice in the list becomes the default.
Prompt *(required)*	Enter the prompt string to display to the user when presenting the choices.

Property	Usage

Field Display Type	The format of the options presented to the user.
	
	Note
	• This property only displays when the node is placed within a Page node.
	• This property is not visible in the AM admin UI. It requires the Platform UI (Advanced Identity Cloud admin console).
	
	Possible values are:
	
	`select`
	Lets the user select one or more options from a selection (default).
	`radio`
	Lets the user select a single option from a group of radio buttons.

Configuration Provider node

The **Configuration Provider** node is a scripted node that dynamically imitates another node and replaces it in the journey.

The script builds a map of configuration properties matching settings for the imitated node. The **Configuration Provider** node uses the settings to imitate the other node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

The specific shared state inputs depend on your script and the configuration it builds. The shared state data must include all required **Script Inputs** properties.

In other words, shared state data must include whatever the **Script** requires to prepare configuration data for the imitated node.

Dependencies

To prepare to use this node:

1. Decide what type of node to imitate.

   The imitated node must have a fixed set of outcomes. You can't use a node type whose outcomes change based on the node configuration.

2. Create an appropriate **Config Provider** script.

   Base your script on the `config-provider-node.js` sample.

3. Obtain the list of required configuration properties for the imitated node.

PingAM

In the AM admin UI, use the API explorer endpoint `/realm-config/authentication/authenticationtrees/nodes/NodeType#_action_template`.

The following request returns the configuration properties for a Message node:

```
$ curl \
--request POST \
--header "<cookie>: <token>" \
"https://openam.example.com:8443/openam/json/realm-config/authentication/authenticationtrees/nodes/
MessageNode?_action=template"
{
  "messageYes": {},
  "message": {},
  "messageNo": {}
}
```

Ping Identity Platform

In the Ping Identity Platform admin UI:

1. Create a journey with the imitated node.
2. Configure the imitated node.
3. Open your browser’s developer tools panel.
4. Select the **Network** tab.
5. Save the journey and examine the body of the REST request JSON to find the configuration of the imitated node.

For example, you find a Message node has these configuration properties:

- `message`
- `messageYes`
- `messageNo`

Your script builds a `config` object, a map of configuration properties matching the settings of the imitated node. The following example consumes the `username` shared state property to build the Message node configuration:

```
config = {
    "message": {"en-GB": `Hi ${nodeState.get("username")}. Please confirm you are over 18.`},
    "messageYes": {"en-GB": "Confirm"},
    "messageNo": {"en-GB": "Deny"},
}
```

Configuration

Property	Usage
Script	Select the script you created for this node.
Node Type	Select the type of node to imitate.
Script Inputs	Optionally limit the shared state data properties in the shared state input to the selected **Script**. Default: `*` (Any available shared state property)

Outputs

The outputs match those of the imitated node.

Outcomes

The **Configuration Provider** node inherits all the outcomes of its configured **Node Type**. Connect these as you would the outcomes of the imitated node.

This node also has a **Configuration failure** outcome. The **Configuration failure** outcome arises when:

- The **Configuration Provider** node failed to build the configuration map.
- The configuration map is missing required values.
- The configuration map is invalid.

Errors

In addition to the messages from the imitated node, this node can log the following:

Warnings

- `Failed to collect inputs of contained node: node-type`

  A required input property was missing.

- `Failed to get outcome provider for node type.`

  The **Node Type** outcomes were missing.

Errors

- `Failed to configure node: node-type`

  This corresponds to the **Configuration failure** outcome.

To troubleshoot HTTP errors this node causes, refer to the *Errors* section of the imitated node.

Examples

In the following example, the **Configuration Provider** node imitates a Message node.

The **Configuration Provider** settings are the following:

Script

A script to configure a Message node dynamically.

The script accesses the `username` from shared state data to set the message:

```
config = {
    "message": {"en-GB": `Hi ${nodeState.get("username")}. Please confirm you are over 18.`},
    "messageYes": {"en-GB": "Confirm"},
    "messageNo": {"en-GB": "Deny"},
}
```

Node Type

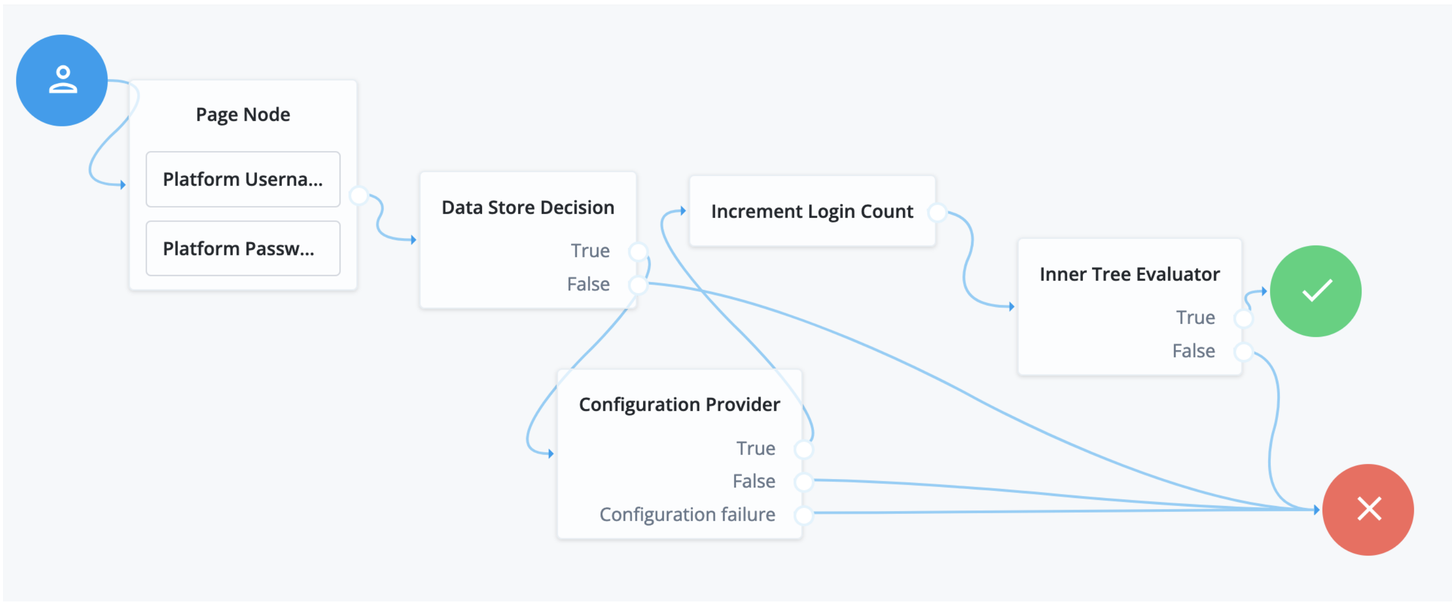
`Message Node`

Script Inputs

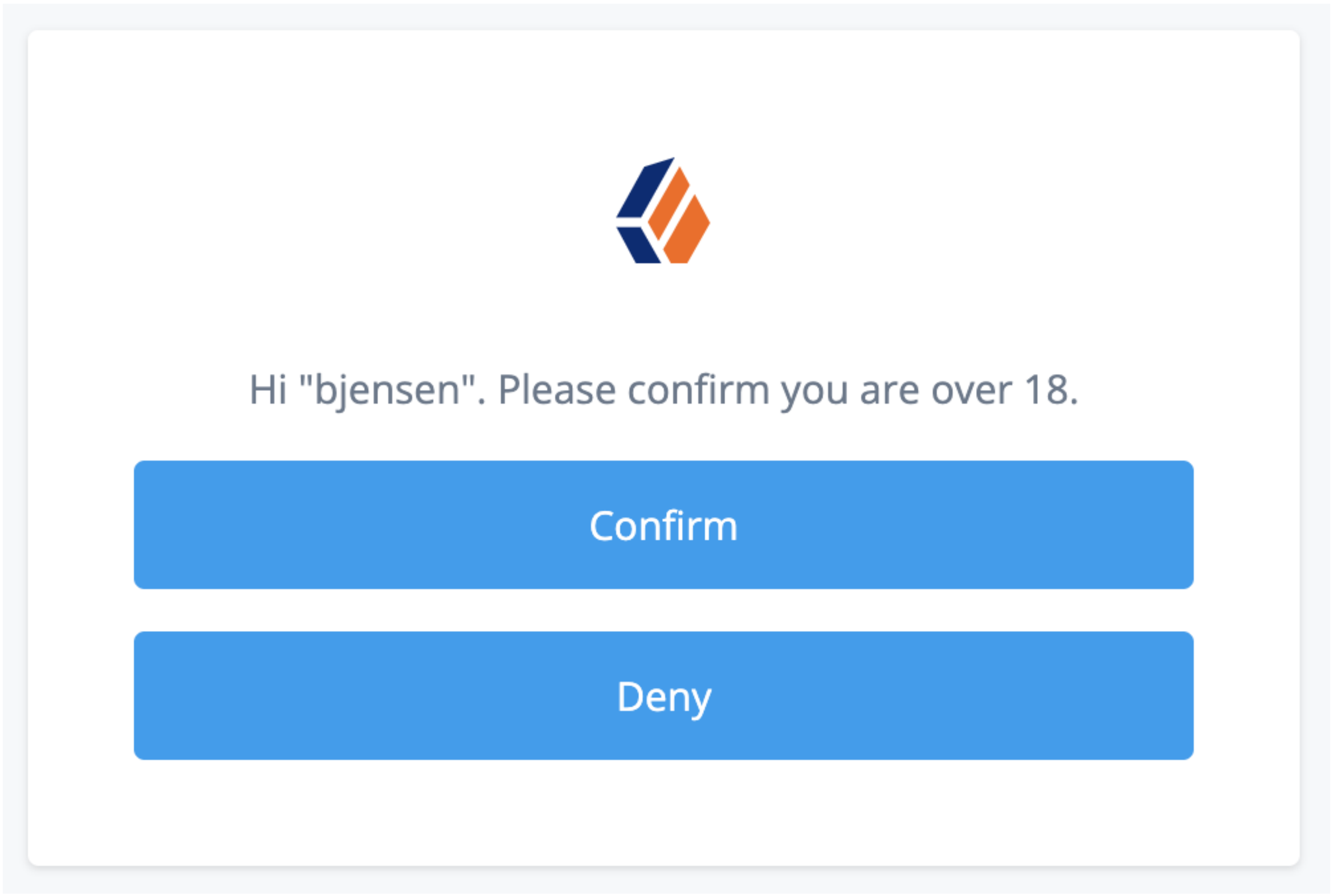
`username`

The default, `*`, also works because `username` is one of the available shared state properties.

The **Configuration Provider** node is part of a journey where the user enters their username and password before getting the message screen, so their username is in the shared state data. Notice the outcomes of the node include those of the Message node (**True**, **False**):



When the journey reaches the **Configuration Provider** node, the script for the node retrieves the `username` and dynamically configures the node. The **Configuration Provider** node, imitating a Message node, prompts the user with the message:


Hi "bjensen". Please confirm you are over 18.

- When the user clicks **Confirm**, the journey continues to the Increment Login Count node.
- When the user clicks **Deny**, the journey continues to the Failure node.
- If the configuration process fails, the node triggers the **Configuration failure** outcome and the journey continues to the Failure node. In this case, you can find the reason for the failure in the logs.

Email Suspend node

The **Email Suspend** node generates and sends an email, such as an address verification email based on an email template. This node relies on the email service configured in IDM to send the email.

This node generates a unique link and passes it as the `resumeURI` property for the template.

The **External Login Page URL** is used as the base of the `resumeURI` if it's been configured. Otherwise, the **Base URL Source** service is used to construct the base of the `resumeURI`.

Learn more in Core authentication attributes > General and Configure the Base URL source service.

Authentication is suspended until the end user clicks the link in the email to resume the flow. Make sure the authentication session is long enough for the end user to complete the flow, so it doesn't time out. Learn more in Configure suspended authentication.

If you don't need to suspend authentication and wait for a reply, use the Email Template node instead.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

The **Email Suspend** node either uses the identity profile in the shared state data or looks up the user profile. In either case, the node uses any applicable profile properties to populate the email template, omitting missing values from the populated template.

If **Object Lookup** is *not* enabled for the node (default), the shared state data must hold the **Email Attribute** with the recipient's email address and any properties the email template uses.

If **Object Lookup** is enabled for the node, the shared state data must hold the profile value to match the configured **Identity Attribute**. The **Email Suspend** node uses the **Identity Attribute** to look up the profile, and its **Email Attribute** to get the recipient's email address from the profile.

Dependencies

Before you use the **Email Suspend** node:

- Configure IDM integration in AM.
- Configure outbound email in IDM.
- Prepare an email template in IDM.

  Record the email template name for use when configuring the **Email Suspend** node.

  Tip

  You can find the email template name in the required format in the URL when you're configuring the template. For example, the **Forgotten Username** email template's name is `forgottenUsername`:

  ```
  https://<tenant-env-fqdn>/?realm=alpha#/email/templates/edit/forgottenUsername
  ```

Configuration

Property	Usage
Email Template Name	The name of the email template prepared as a dependency. Default: `registration`
Email Attribute	The shared state data property or profile attribute for the recipient's email address. Default: `mail`
Email Suspend Message	The localized message to display when the node suspends authentication. According to OWASP authentication recommendations, the message should be the same regardless of the validity of the recipient's email address. You can use plain text or HTML code in this message. Default: `An email has been sent to the address you entered. Click the link in that email to proceed.`
Object Lookup	Whether to look up the managed identity profile. Default: disabled
Identity Attribute	The attribute used to identify the managed object in IDM. The node uses this when **Object Lookup** is enabled. Default: `userName`

Outputs

This node doesn't add to the shared state data.

Outcomes

The **Email Suspend** node has a single outcome path.

Evaluation continues when the end user clicks the link in the email to resume the flow.

Errors

This node doesn't log any error or warning messages of its own.

Examples

The following default journeys use the **Email Suspend** node:

- `ForgottenUsername`
- `ResetPassword`
- `UpdatePassword`

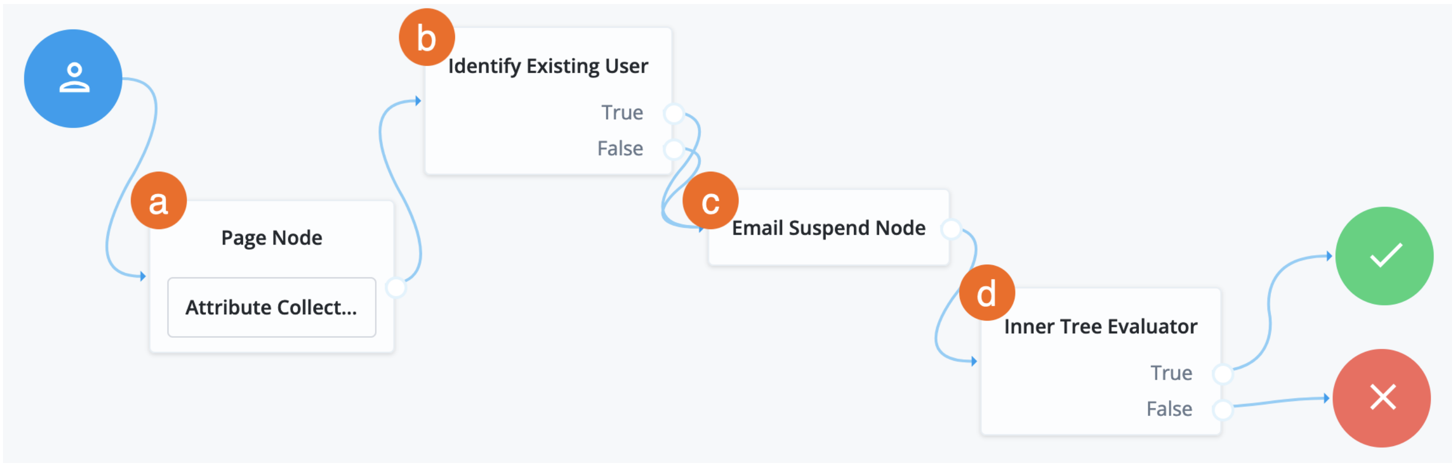
Forgotten username

In the default journey for recovering a forgotten username, the end user enters their email address to recover their username.

Before you start

- Configure the email service.
- Optionally use the email template editor to modify the `forgottenUsername` template.

The journey

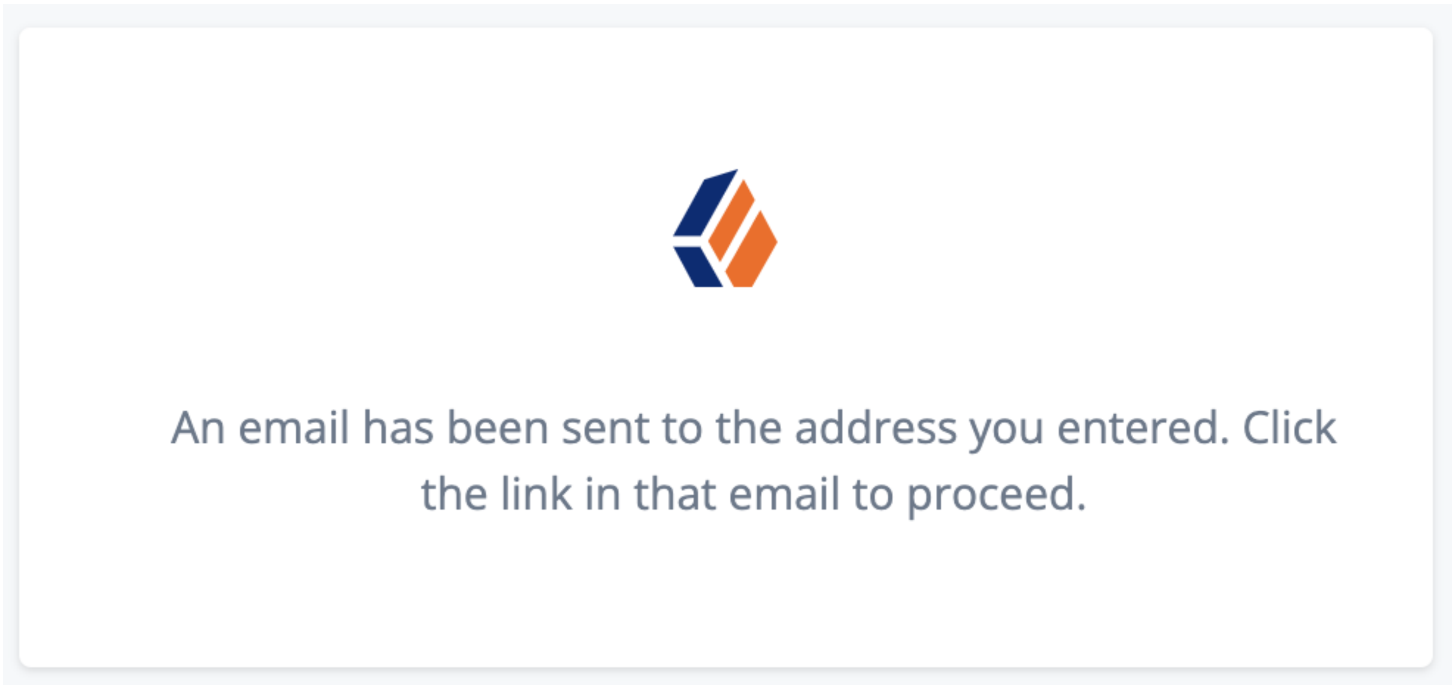


a The Page node with an Attribute Collector node prompts for the end user's email address.

b The Identify Existing User node attempts to look up the username by matching the email address to the email address in an identity profile.

The lookup fails if more than one user profile uses the same email address.

c The **Email Suspend** node reads the user profile, generates a unique `resumeURI` link to resume the journey, and populates the `forgottenUsername` email template. On success, the node makes a request to the email service to send the email. In any case, it displays the suspend message:



An email has been sent to the address you entered. Click the link in that email to proceed.

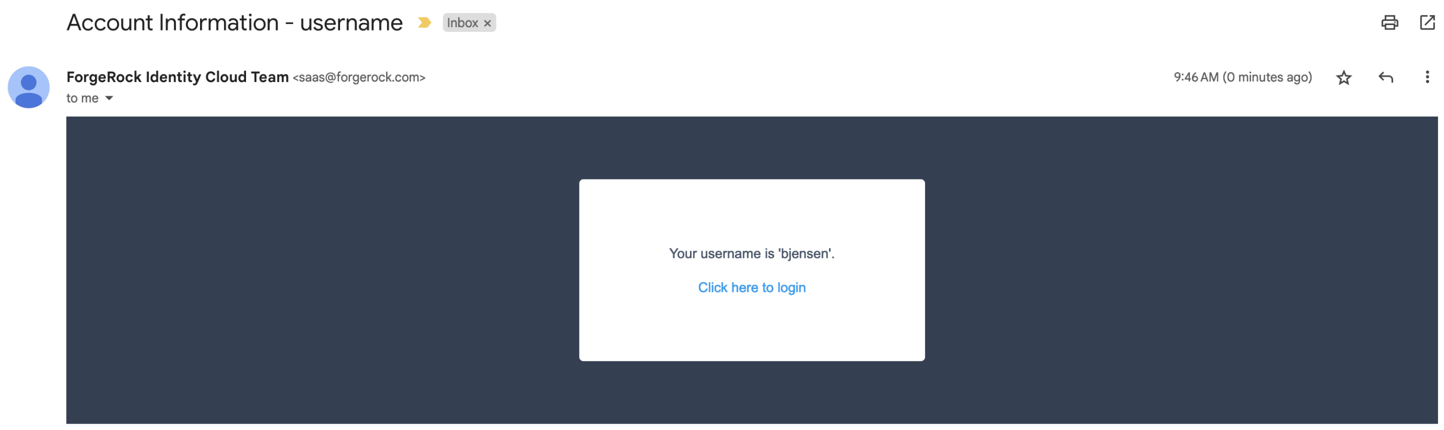
The node's settings are:

Email Template Name	`forgottenUsername`
Email Attribute	`mail` (default)
Email Suspend Message	`An email has been sent to the address you entered. Click the link in that email to proceed.` (default)
Object Lookup	Enabled
Identity Attribute	`mail`

d When the end user clicks the link to resume the journey, the Inner Tree Evaluator node starts the `Login` journey.

Try the journey

Use the journey to recover the username for an account whose email you have access to. For example, if Babs Jensen's account has your email address, the **Email Suspend** node sends you a message such as the following:



Account Information - username Inbox ×

ForgeRock Identity Cloud Team <saas@forgerock.com>
to me

9:46 AM (0 minutes ago)

Your username is 'bjensen'.

Click here to login

Follow the link to continue the journey and log in as Babs Jensen.

Registration

For an example registration journey showing how to use the **Email Suspend** node and the Email Template node, refer to the **Email Template** node examples.

Email Template node

The **Email Template** node generates and sends an email, such as a welcome email based on an email template. This node relies on the email service configured in IDM to send the email.

This node doesn't wait for a reply. If authentication should pause and wait for a reply to the email, use the Email Suspend node instead.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed) **Note** This functionality requires that you configure AM as part of a sample Ping Identity Platform deployment.	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

The **Email Template** node uses the identity in the shared state data to get the profile, meaning the journey must have successfully authenticated or at least identified the recipient. When the journey reaches this node, the shared state must hold the profile value to match the configured **Identity Attribute**. The value can be in the `username` property or in a property having the same name as the **Identity Attribute**.

The **Email Template** node uses its **Identity Attribute** to look up the profile, and its **Email Attribute** to get the recipient's email address from the profile. In other words, the node finds the recipient's address and other properties in the *profile*, not the shared state data.

For example, if the node uses default configuration settings and Babs Jensen authenticated to the journey, the shared state includes `"username": "bjensen"`. The node looks for a profile with `"userName": "bjensen"`. It gets the recipient address from the profile's `mail` attribute, such as `"mail": "bjensen@example.com"`. The node uses any applicable profile attributes to populate the email template, omitting missing values from the populated template.

Dependencies

Before you use the **Email Template** node:

- Configure IDM integration in AM.
- Configure outbound email in IDM.
- Prepare an email template in IDM.

  Record the email template name for use when configuring the **Email Template** node.

> **Tip**
>
> You can find the email template name in the required format in the URL when you're configuring the template. For example, the **Forgotten Username** email template's name is `forgottenUsername`:
>
> ```
> https://<tenant-env-fqdn>/?realm=alpha#/email/templates/edit/forgottenUsername
> ```

Configuration

Property	Usage
Email Template Name	The name of the email template prepared as a dependency. Default: `welcome`
Email Attribute	The profile attribute for the recipient's email address. Default: `mail`
Identity Attribute	The attribute used to identify the managed object in IDM. Default: `userName`

Outputs

This node doesn't add to the shared state data.

If the outcome is `Email Sent`, this node has sent the templated message to the recipient through the email service.

Outcomes

`Email Sent`

The node completed a request to send the message to the recipient.

If the message doesn't reach its destination, the problem is with the delivery, not with the node.

`Email Not Sent`

The node failed to complete a request to send the message.

This outcome arises, for example, when one of the following happens:

- The node can't get the user profile.
- The template doesn't match the user profile.
- The specified **Email Attribute** doesn't contain an address.

According to OWASP authentication recommendations, any messages displayed in the journey should be the same in both cases.

Errors

This node doesn't log any error or warning messages of its own.

Examples

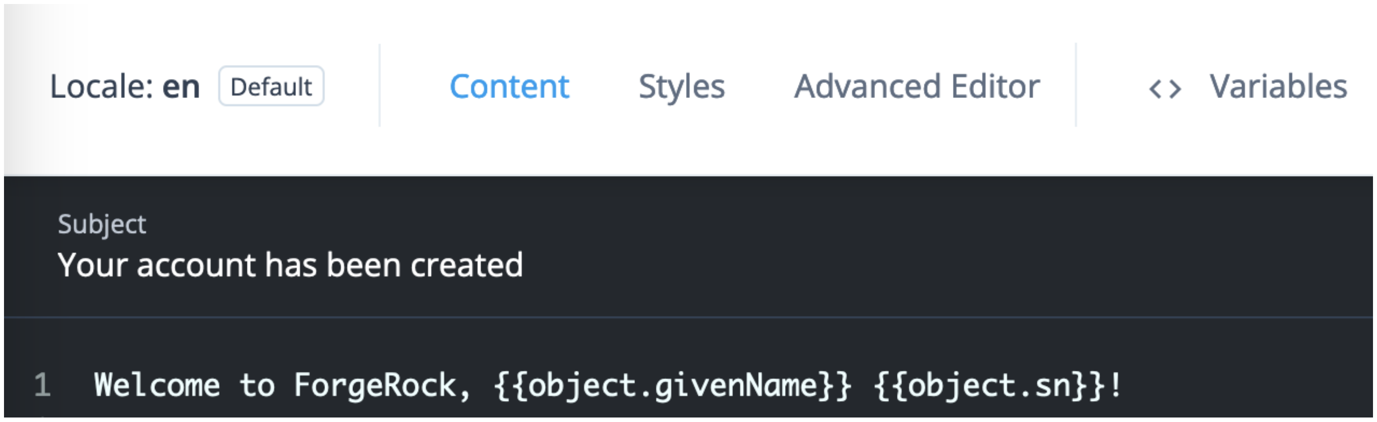
Use the **Email Template** node to send an email message when the journey doesn't depend on a reply. For example, send a welcome message when a user completes registration.

This example augments the default **Registration** journey and sends a welcome email.

Before setting up the journey:

- Configure the email service.
- Create an email template for the **Email Template** node.

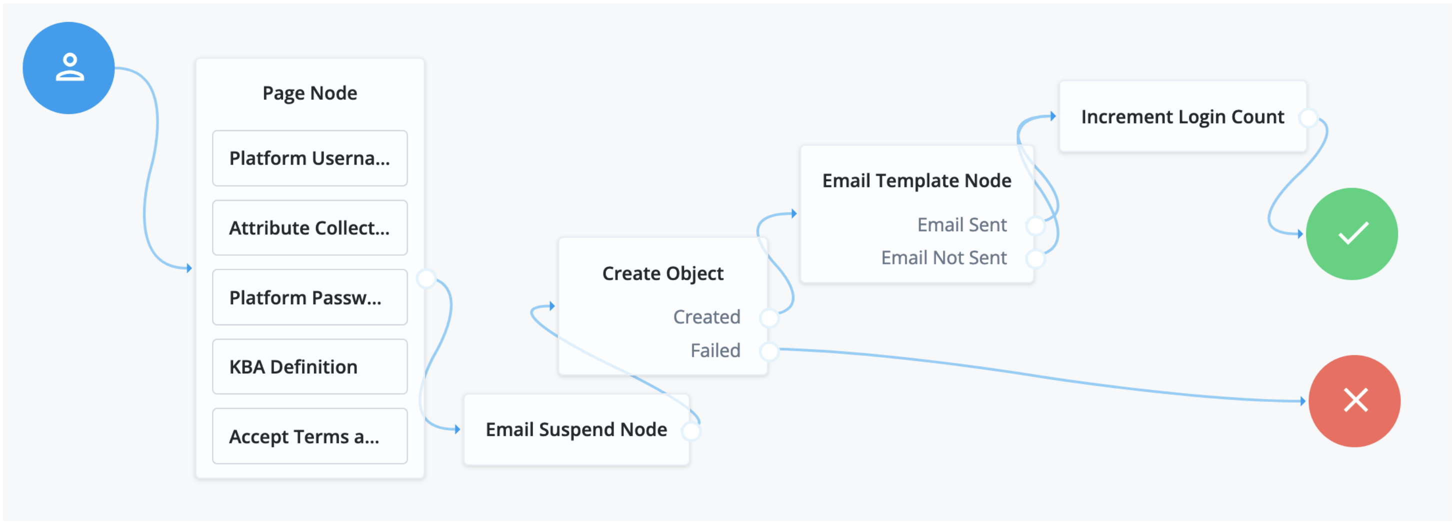
  Use the platform email template editor to duplicate the `Welcome` template and customize your copy:



Subject
Your account has been created

```
Welcome to ForgeRock, {{object.givenName}} {{object.sn}}!
```

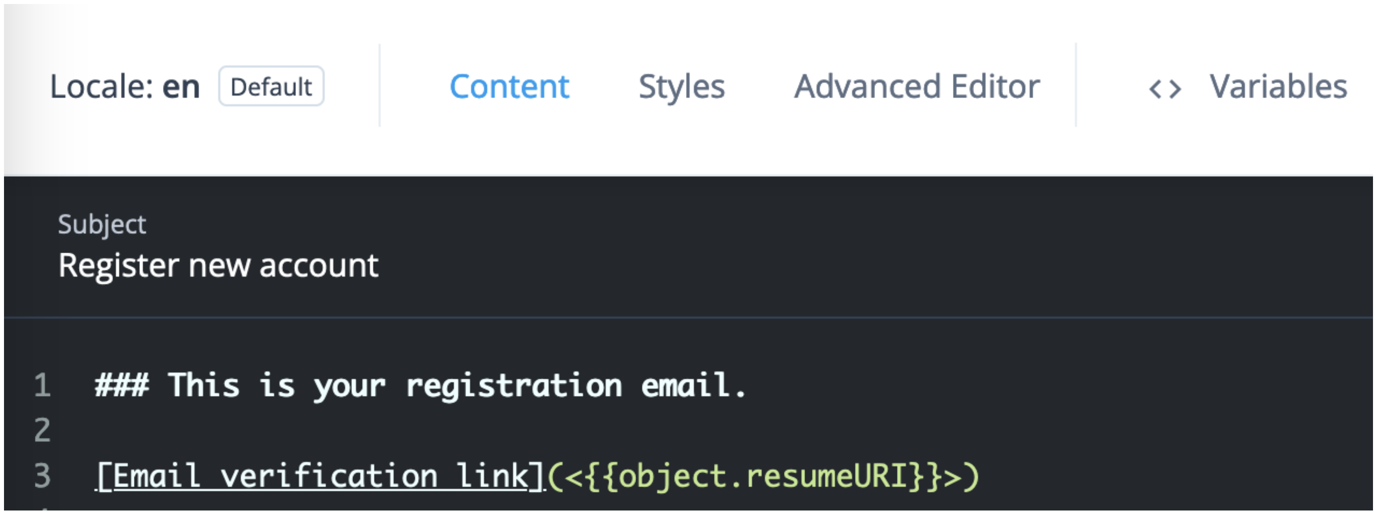
The journey is as follows:



1. The Page node prompts for the same information as the default **Registration** journey.
2. The Email Suspend node sends a message to the registered email address with a link for the user to click.

   The journey proceeds when the user clicks the link, confirming their email address.

   It has default settings and uses the default `Registration` email template:



Subject
Register new account

```
### This is your registration email.

[Email verification link](<{{object.resumeURI}}>)
```

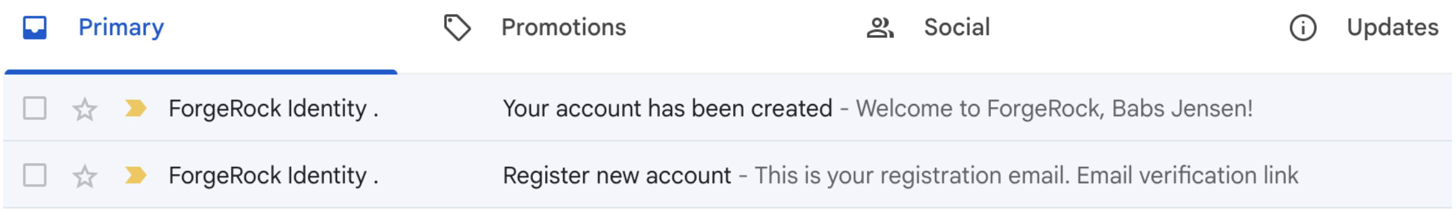
3. The Create Object node stores the newly registered user’s profile.
4. The **Email Template** node reads the user profile and populates the template from profile attribute values. It makes a request to the email service to send the message.

   Its settings are:

Email Template Name	The name of your welcome email template
Email Attribute	`mail` (default)
Identity Attribute	`userName` (default)

5. The Increment Login Count node updates the count on successful authentication.

Use the journey to register an account for Babs Jensen with your email as the address. You receive two messages:



ForgeRock Identity . Your account has been created - Welcome to ForgeRock, Babs Jensen!
ForgeRock Identity . Register new account - This is your registration email. Email verification link

- The `Register new account` message has a link to click to continue the journey, confirming you can access the registered email account.
- The `Your account has been created` welcomes you on successful registration.

This demonstrates you have successfully used the **Email Template** node in a journey.

Failure URL node

Sets the redirect URL when authentication fails.

Note

Specifying a failure URL overrides any `gotoOnFail` query string parameters.

Learn more about how AM determines the redirection URL, and configuring the Validation Service to trust redirection URLs, in Configure success and failure redirection URLs.

Tip

The URL is also saved in the shared `nodeState` object on the `failureUrl` key.
Find more information in Customize authentication trees.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

Single outcome path.

Properties

Property	Usage
Failure URL *(required)*	Specify the full URL to redirect to when authentication fails.

Get Session Data node

The **Get Session Data** node retrieves the value of a specified key from a user's session data, and stores it in the specified key of the shared state (in scripts, the `nodeState` object).

Use this node only during session upgrade—when the user has already successfully authenticated previously and is now upgrading their session for additional access. Learn more in Session upgrade.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node reads values from the user's session data.

Dependencies

This node can complete its function only when the user has an existing session. Precede this node in the flow with a Scripted Decision node using a script to determine whether an existing session is present:

```
if (typeof existingSession !== 'undefined') {
  outcome = "hasSession";
} else {
  outcome = "noSession";
}
```

Configuration

All the configuration properties are required:

Property	Usage
Session Data Key	Specify the session data key whose value the node reads. Default: none
Shared State Key	Specify the name of the shared node state field to hold the session data. Default: none

Outputs

This node writes the **Session Data Key** value in the **Shared State Key** field of the shared node state.

It also writes the field and its value to the `objectAttributes` object in the shared node state.

Outcomes

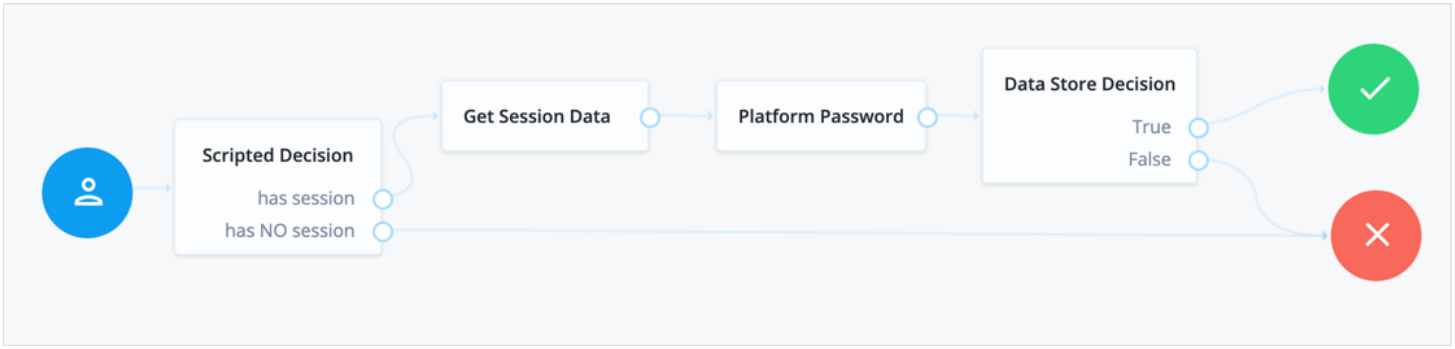
Single outcome path; on success, the **Shared State Key** in the shared node state holds the session data.

Errors

If it cannot read the **Session Data Key** value, this node logs an `Exception occurred trying to get data (<session-data-key>) from existing session` error message.

Example

When the user has an active session, the following example gets the username from the session, collects the password, and confirms the username-password credentials:



The following table includes example keys from an existing session with their corresponding sample values:

Key	Sample value
`AMCtxId`	`e370cca2-02d6-41f9-a244-2b107206bd2a-122934`
`amlbcookie`	`01`
`authInstant`	`2023-04-04T09:19:05Z`
`AuthLevel`	`0`
`CharSet`	`UTF-8`
`clientType`	`genericHTML`
`FullLoginURL`	`/am/XUI/?realm=alpha#login/`
`Host`	`34.117.172.39`
`HostName`	`am.forgeblocks.com`
`Locale`	`en_US`

Key	Sample value
`Organization`	`dc=openam,dc=forgerock,dc=org`
`Principal`	`uid=amAdmin,ou=People,dc=openam,dc=forgerock,dc=org`
`Principals`	`amAdmin`
`Service`	`ldapService`
`successURL`	`/openam/console`
`sun.am.UniversalIdentifier`	`uid=amAdmin,ou=People,dc=openam,dc=forgerock,dc=org`
`UserId`	`amAdmin`
`UserProfile`	`Required`
`UserToken`	`amAdmin`
`webhooks`	`myWebHook`

Inner Tree Evaluator node

The **Inner Tree Evaluator** node lets you nest authentication journeys as children within a parent. There is no limit to the depth of nesting.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

Any information collected or set by the parent journey, such as a username or the authentication level, is available in the child journeys.

For information about shared state data, refer to Access shared state data.

Dependencies

None.

Configuration

Property	Usage
Tree Name	Select or enter the name of the authentication journey to evaluate. You must set this value; there is no default.

Outputs

Shared node state data collected by child journeys is available to the parent when evaluation of the child is complete, but data stored in transient and secure state is not. For instance, if a child journey collects and stores the user's password in transient state, it cannot be retrieved by a node in the parent journey when evaluation continues.

For information about shared state data, refer to Access shared state data.

Outcomes

`True`

Successfully reached the `Success` node of the child.

`False`

Any other case.

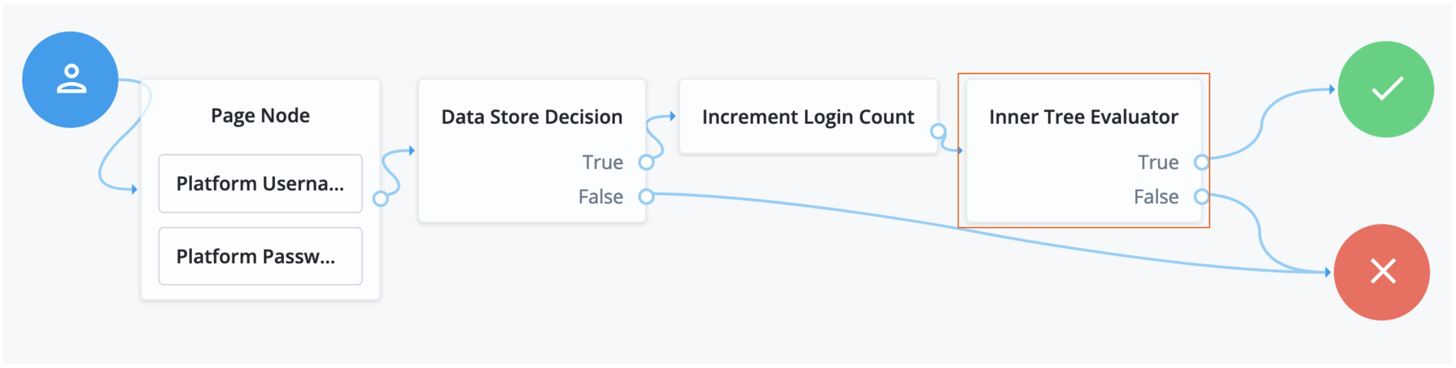
Errors

If it cannot get the shared node state from the child, this node logs an `Exception when gathering inner tree inputs` message.

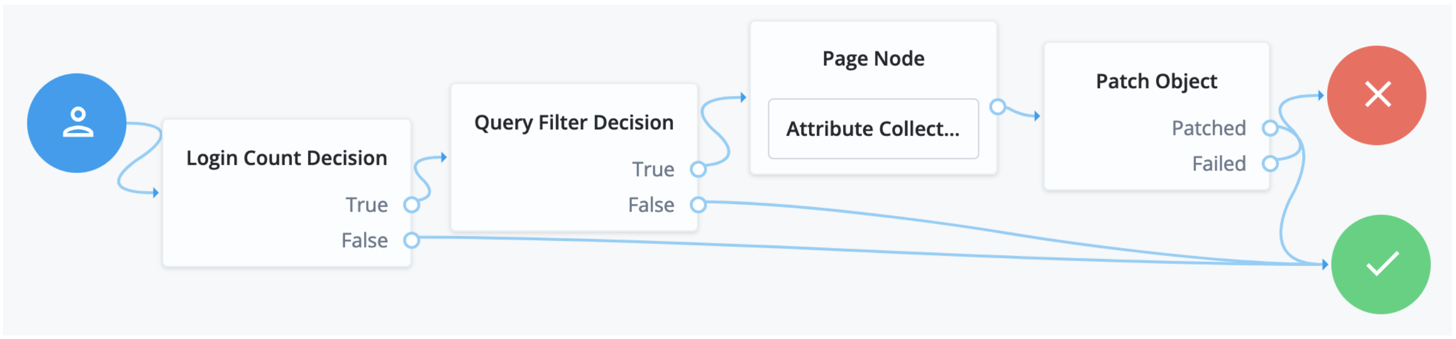
If the **Tree Name** doesn't match an existing journey, this node throws an exception with a `Configured tree does not exist: <tree-name>` message.

Example

The following journey uses an Inner Tree Evaluator node for progressive profiling:



- The Page node presents a page with input fields to prompt for the username and password.
  - The Platform Username node collects and injects the `userName` into the shared node state.
  - The Platform Password node collects and injects the `password` into the shared node state.
- The Data Store Decision node uses the username and password to determine whether authentication is successful.
- The Increment Login Count node updates the login count on successful authentication.
- The Inner Tree Evaluator node (outlined) invokes a nested journey:



- The Login Count Decision node triggers the rest of the journey depending on the login count and its settings.
- The Query Filter Decision node determines whether managed object profile fields are still missing.
- The Attribute Collector node in the Page node requests additional input for the profile.
- The Patch Object node stores the additional input in the managed object profile.

Message node

The **Message** node presents a custom, localized message to the user with customizable, localized positive and negative answer buttons the user must click to proceed.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node reads preferred locales from the incoming request context.

It doesn’t read from the shared node state.

This node has no required predecessor nodes.

Dependencies

None.

Configuration

Property	Usage
Message	Add a custom, localized message per locale: 1. Click **Add**. 2. In the **Key** field, enter the locale.(1) The incoming HTTP request can include an `Accept-Language` header indicating the user’s preferred locales. If the incoming HTTP request doesn’t include the header or the preferred locales don’t match any configured locales, the node uses default settings. It uses the **Realms** > ***Realm Name*** > **Authentication** > **Settings** > **General** > **Default Authentication Locale** setting from the AM admin UI. If there is no default authentication locale, the node uses **Deployment** > **Servers** > ***Server Name*** > **General** > **System** > **Default Locale**. 3. In the **Value** field, enter the message to display to the user. If you leave this blank, the message node displays a localized version of `Default message` to the user. To edit an entry, click the **Pencil** icon (✎). To remove an entry, click the **Delete** icon (🗑).

Property	Usage
Positive answer	Add the text per locale for the positive answer button that triggers a **True** outcome: 1. Click **Add**. 2. In the **Key** field, enter the locale.(1) If the incoming HTTP request doesn't include the header or the preferred locales don't match any configured locales, the node uses the first text in the list. 3. In the **Value** field, enter the text to display to the user. If you leave this blank, the button displays a localized version of `Yes`. To edit an entry, click the **Pencil** icon (✎). To remove an entry, click the **Delete** icon (🗑).
Negative answer	Add the text per locale for the negative answer button that triggers a **False** outcome: 1. Click **Add**. 2. In the **Key** field, enter the locale.(1) If the incoming HTTP request doesn't include the header or the preferred locales don't match any configured locales, the node uses the first text in the list. 3. In the **Value** field, enter the text to display to the user. If you leave this blank, the button displays a localized version of `No`. To edit an entry, click the **Pencil** icon (✎). To remove an entry, click the **Delete** icon (🗑).
Shared State Property Name	The name of the node state property. If set, the node adds the property to shared node state, setting its value to the numeric value of the outcome: ***0*** The user clicked the positive answer button. ***1*** The user clicked the negative answer button. For example, if you set this to `messageNodeOutcome` and the user clicks the positive answer button, the node adds `"messageNodeOutcome": 0` as a shared node state property.
Only Positive Answer	(Ping Identity Platform deployments only) When enabled, the node displays only the positive answer button. This property is available only in the Ping Identity Platform admin UI. ⓘ **Note** This property only displays when the node is placed within a Page node in a Ping Identity Platform deployment.

Property	Usage
Show buttons as links	When enabled, the node shows the buttons as links instead. This property is available only in the Ping Identity Platform admin UI. **Note** This property only displays when the node is placed within a Page node in a Ping Identity Platform deployment.

(1) Specify a locale that Java supports, such as `en-gb`. Otherwise, the node throws a configuration exception with an `Invalid locale provided` message.

Outputs

When the **Shared State Property Name** setting has a value, the node adds the property to the shared node state. The property's value is the numeric value of the outcome:

`0`

The user clicked the positive answer button.

`1`

The user clicked the negative answer button.

Outcomes

Returns a boolean outcome:

True

The user clicked the positive answer button.

False

The user clicked the negative answer button.

Errors

This node doesn't cause authentication to fail unless you connect one of the outcomes to a Failure node.

If the message or answer button settings specify a locale Java doesn't support, the node throws a configuration exception with an `Invalid locale provided` message. If this happens, fix the locale setting.

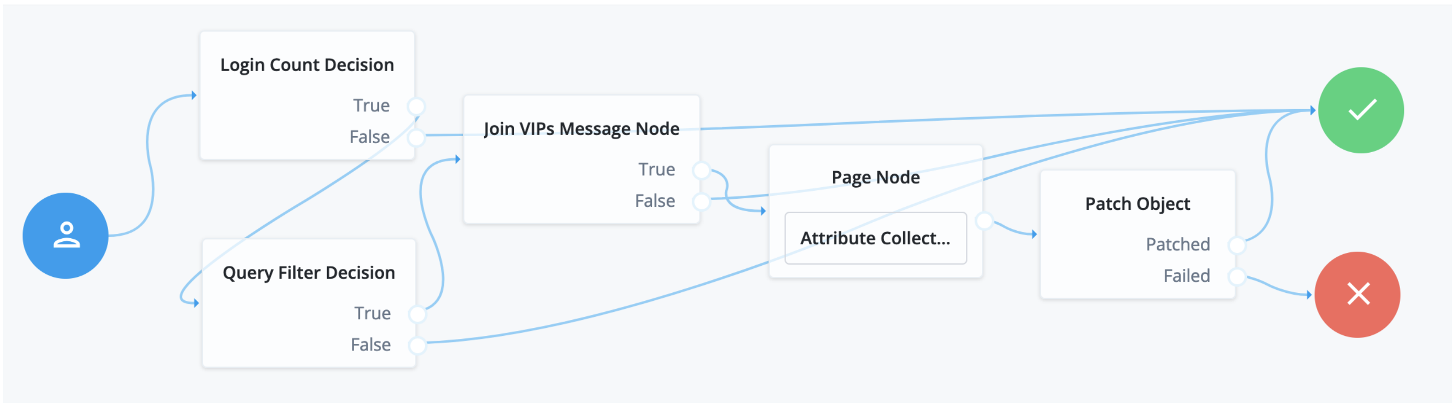
If this node encounters an internal configuration issue, it logs a warning message `Error attempting to retrieve the realm/global default locale`. If the warning persists, contact Ping Identity Support.

Examples

Use a **Message** node to:

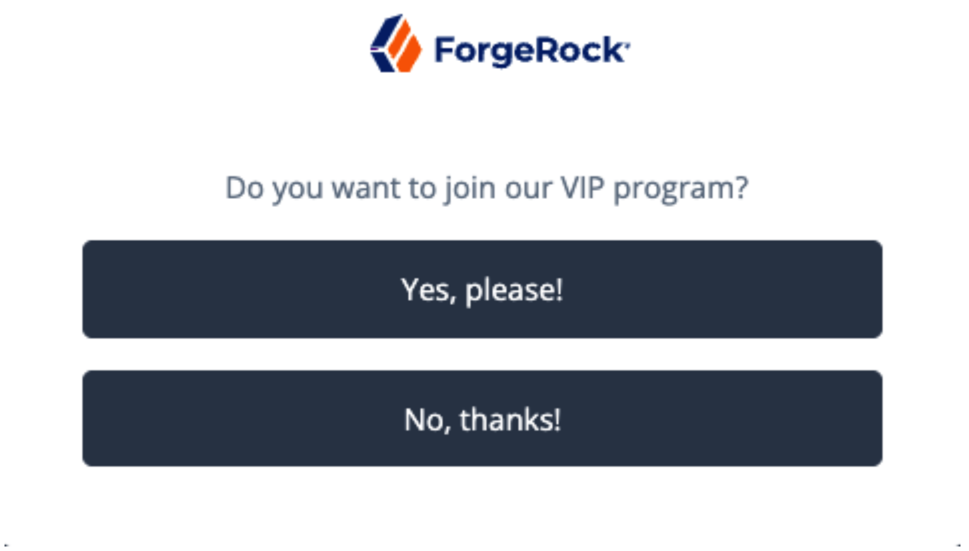
- Communicate an important message for the user to acknowledge.
- Ask a question with a yes/no answer.

The following journey uses the `Join VIPs Message Node` to prompt the user to join the VIP program:



- The Login Count Decision node triggers the Query Filter Decision node after every tenth authentication.
- The Query Filter Decision node queries the user profile to determine whether they have signed up for the VIP program.

  If the user hasn't signed up yet, the **True** outcome triggers the `Join VIPs Message Node`, which prompts the user to join the program:



Node property settings:

Message

`en-gb` ; `Do you want to join our VIP program?`

Positive answer

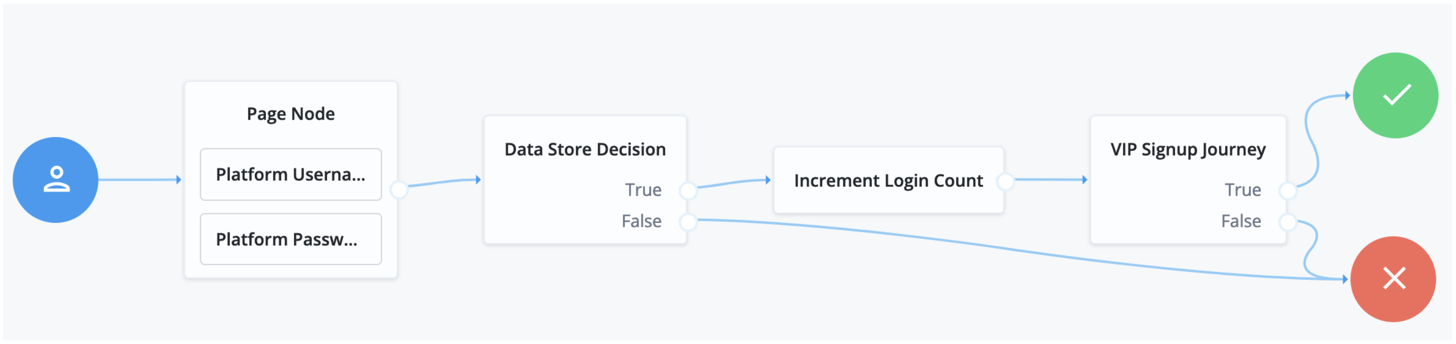
`en-gb` ; `Yes, please!`

Negative answer

`en-gb` ; `No, thanks!`

- If the user clicks **Yes, please!** the Page node with an embedded Attribute Collector node collects opt-in choices to store in user profile attributes.
- The Patch Object node updates the user profile with the attributes collected.

Call the journey using an Inner Tree Evaluator node from another authentication journey directly after an Increment Login Count node note:



The `VIP Signup Journey` uses the login count from the Increment Login Count node in the Login Count Decision node to decide whether to prompt the user to join the VIP program.

Meter node

Increments a specified metric key each time evaluation passes through the node.

Learn more about the `Meter` metric type in Monitoring metric types. The metric is exposed in all available interfaces, as described in Monitor AM instances.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

Single outcome path.

Properties

Property	Usage
Metric Key *(required)*	Specify the name of a metric to increment when evaluation passes through the node. Example: `authentication.success` For the list of available metrics, refer to Monitoring metrics.

Page node

The **Page** node lets you combine multiple nodes that request input onto a single page for display to the user.

Drag and drop nodes onto the Page node to combine them. Only add nodes that use callbacks to request input. Don't add other nodes, such as the Data Store Decision node and the Push Sender node to this node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

The inputs are determined by the collective inputs of the contained nodes.

Dependencies

None.

Configuration

Property	Usage
Page Header	Optional. A localized title for the Page node and the nodes contained within it. Use this when components of an authentication journey need a title; for example, dividing a registration flow into labeled sections.

Property	Usage
Page Description	Optional. A localized description for the Page node and the nodes contained within it. Use this when you need additional descriptive text in an authentication journey. In a Ping Identity Platform deployment, you can use HTML code to format the description.
Stage	An optional stage name to pass to the client to aid in rendering.
Submit Button Text	Optional. Use the **Key** and **Value** fields to set the text of the **Submit** button. This property is available only in the Ping Identity Platform admin UI.
Page Footer	Optional. A localized footer for the page node and the nodes contained within it. Use this when you need additional descriptive text in an authentication journey. In a Ping Identity Platform deployment, you can use HTML code to format the footer. This property is available only in the Ping Identity Platform admin UI.
Theme	Optional. If using hosted pages, specify a theme to override this journey's UI theme. This property is available only in the Ping Identity Platform admin UI.

Note

This node's optional properties are passed in the response, but a self-hosted or custom UI must support these properties to make them visible to the end user.

Outputs

The outputs are determined by the collective outputs of the contained nodes.

Outcomes

The outcomes are determined by the last node in the Page node. Only the last node in the page can have more than one outcome path.

Errors

This node can log the following error messages:

Message	Notes
`Failed to collect inputs of contained node: <node-name>`	The <node-name> could not retrieve required properties from the shared node state
`Failed to collect outputs of contained node: <node-name>`	The <node-name> could not retrieve required properties to include in the shared node state

Message	Notes
`Could not find the identity based on the information available on context`	Failed to find the account profile with this `username` in this `realm`
`An error occurred when trying to lock out the user account`	Failed to update the account status; applies when locking and unlocking the account

This node can throw exceptions with the following messages during operation:

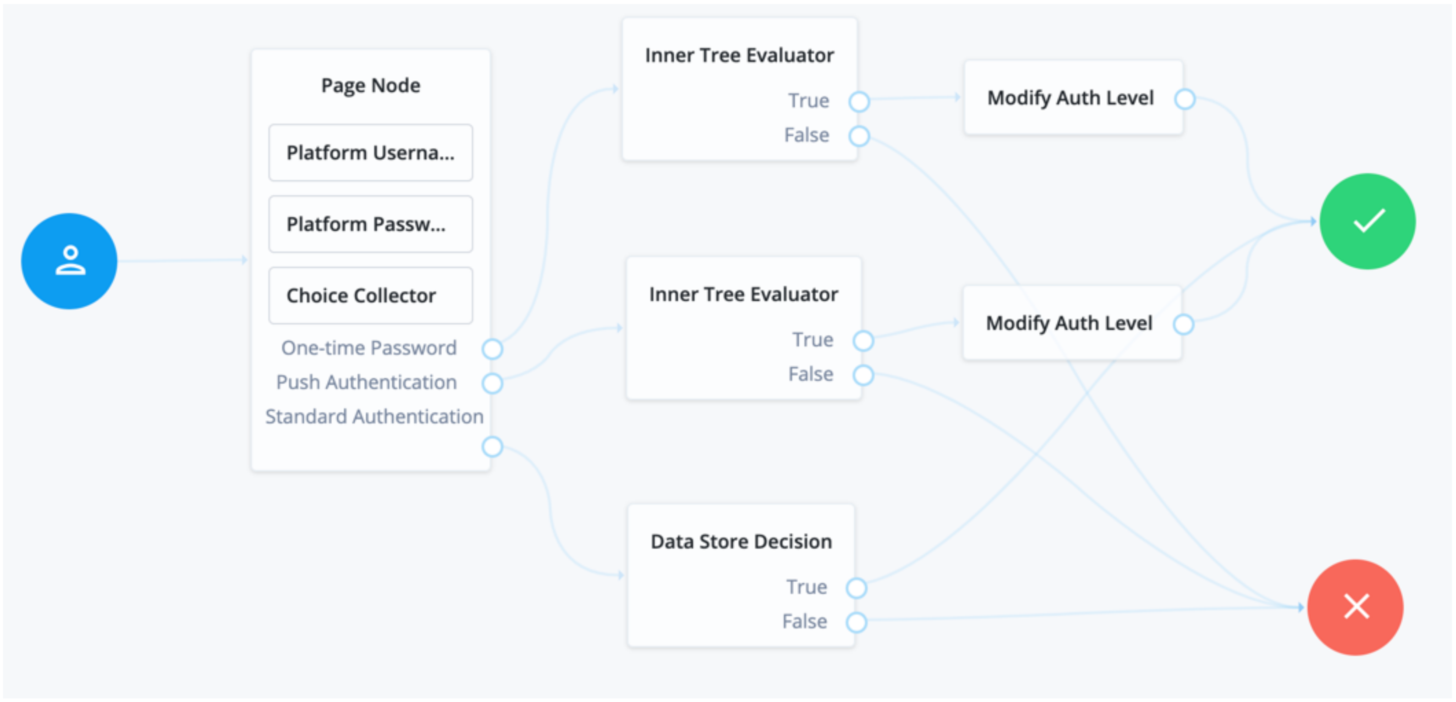
Message	Notes
`This page has no nodes in it, so cannot proceed`	A Page node must contain at least one other node
`No outcome and only metadata callbacks found`	Failed to get to an outcome while processing the contained nodes
`Node properties cannot be fetched`	Failed to access the properties of a contained node

This node can throw exceptions with the following messages when saving the journey:

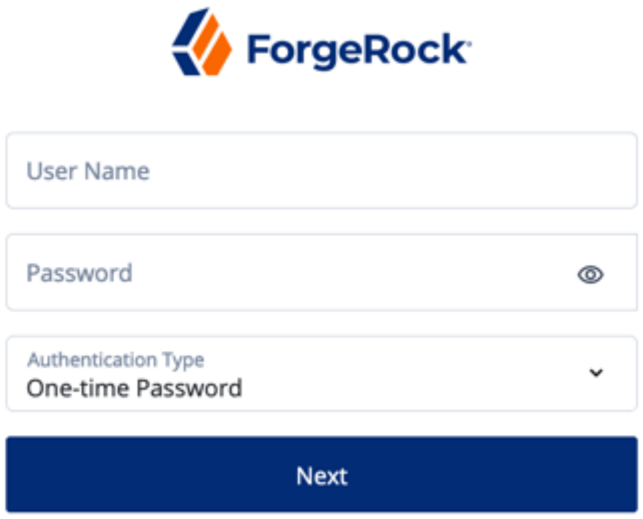
Message	Notes
`Illegal child node type: <node-type>`	A Page node can't contain a <node-type>
`Node does not have any outcomes: <node-type>`	The contained nodes must have at least a single outcome path
`Only the last node in a page can have more than one outcome`	Consider rearranging the contained nodes
`Node does not exist: <node-id>`	Use the journey editor to fix the problem
`Could not load child node: <node>`	Use the journey editor to fix the problem
`Could not obtain outcomes for node: <node>`	Use the journey editor to fix the problem

Example

The following example uses a Page node containing a Platform Username node, Platform Password node, and Choice Collector node:



The flow prompts the user for all input on a single page:



Polling Wait node

Pauses authentication progress for a specified number of seconds, for example, to wait for a response to a one-time password email or push notification.

Requests made during the wait period are sent a `PollingWaitCallback` callback and an authentication ID. For example, the following callback indicates a wait time of 10 seconds:

```
{
    "authId": "eyJ0eXAiOiJK...u4WvZmiI",
    "callbacks": [
        {
            "type": "PollingWaitCallback",
            "output": [
                {
                    "name": "waitTime",
                    "value": "10000"
                },
                {
                    "name": "message",
                    "value": "Waiting for response..."
                }
            ]
        }
    ]
}
```

The client must wait 10 seconds before returning the callback data, including the `authId`:

```
$ curl \
--request POST \
--header "Accept-API-Version: resource=2.0, protocol=1.0" \
--header "Content-Type: application/json" \
--data '{
  "authId": "eyJ0eXAiOiJK…u4WvZmiI",
  "callbacks": [
      {
          "type": "PollingWaitCallback",
          "output": [
              {
                  "name": "waitTime",
                  "value": "10000"
              },
              {
                  "name": "message",
                  "value": "Waiting for response…"
              }
          ]
      }
  ]
}' \
'https://am.example.com:8443/am/json/realms/root/realms/alpha/authenticate?
authIndexType=service&authIndexValue=Example'
```

The end user UI automatically waits for the required amount of time and resubmits the page to continue evaluation. The message displayed during the wait is configurable with the **Waiting Message** property.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `Done`
- `Exited` (configurable)
- `Spam` (configurable)

Evaluation continues along the `Done` outcome path when the next request is received after the wait time has passed.

Enabling Spam detection adds a `Spam` outcome path to the node. Evaluation continues along the `Spam` outcome path if more than the specified number of requests are received during the wait time.

Enabling the user to exit without waiting adds an `Exited` outcome path to the node. Evaluation continues along the `Exited` outcome path if the user clicks the button that appears when the option is enabled. The message displayed on the exit button is configurable by using the **Exit Message** property.

Properties

Property	Usage
Seconds To Wait	Specify the number of seconds to pause authentication. Default: `8`
Enable Spam Detection	Specify whether to track the number of responses received during the wait time, and continue evaluation along the `Spam` outcome path if the number specified in the **Spam Tolerance** property is exceeded. Default: Disabled
Spam Tolerance	Specify the number of responses to allow during the wait time before continuing evaluation along the `Spam` outcome path. This property only applies if spam detection is enabled. Default: `3`

Property	Usage
Waiting Message	Specifies the optional message to display to the user. Provide the message in multiple languages by specifying the locale in the **KEY** field. For example, `en-gb` .[1] The locale selected for display is based on the user's locale settings in their browser. Leave blank to use the default message.[2]
Exitable	Whether the user can exit the node during the wait period. Enabling this option adds a button with a configurable message to the page. Clicking the button causes evaluation to continue along the `Exited` outcome path. Default: Disabled
Exit Message	The optional message to display to the user on the button used to exit the node before the wait period has elapsed. For example, `Cancel` or `Lost phone? Use Recovery Code`. This property only applies if the **Exitable** property is enabled. Provide the message in multiple languages by specifying the locale in the **KEY** field. For example, `en-gb`.[1] The locale selected for display is based on the user's locale settings in their browser. Leave blank to use the default message.[2]

[1] Specify a locale that Java supports, such as `en-gb`. Otherwise, the node throws a configuration exception with an `Invalid locale provided` message.

[2] PingAM only: Learn more about customizing and translating default messages in Internationalize nodes.

Query Parameter node

The **Query Parameter** node lets you insert query parameter values from a journey URL into configurable node state properties. This lets you customize journeys based on the query parameter values.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node takes its inputs from the journey URL and maps each query parameter to a property in the node state. For multi-valued parameters, the node delimits the parameters by a comma (`,`) symbol. In this case, the value assigned to the node state property is a comma-delimited list of items.

The node doesn't read input from the state.

The node has no required predecessor nodes.

Dependencies

None.

Configuration

Property	Usage
Allowed query parameters	List the query parameters the node can obtain from the URL and map them to a property in the node state: 1. Click **Add**. 2. In the **Key** field, enter the query parameter name. 3. In the **Value** field, enter the node state property that will hold the value of this query parameter. This field sets an *allowlist* of parameters the node can pull into the node state. Exercise caution when you create this list to avoid injecting harmful data into the node state. If a query parameter in the URL isn't in this list, the node ignores it. To edit an entry, click the **Pencil** icon (✎). To remove an entry, click the **Delete** icon (🗑).
Allowed query parameters to be delimited	Specify the allowed query parameters that can take multiple values and whose values you want to store in the node state in a comma-delimited list; for example, `["yellow", "green", "red"]`. Enter the query parameter name in the **Add value** field and click **Add**. If you don't delimit the values of a multi-valued query parameter, the node stores the values in the node state as a single string value; for example, `["yellow, green, red"]` . To edit an entry, click the **Pencil** icon (✎). To remove an entry, click the **Delete** icon (🗑).

Outputs

- If the **Allowed query parameters** setting has one or more values, the node adds the values of the listed URL parameters to the corresponding properties in the node state.
- If the **Allowed query parameters** setting has a value but that query parameter isn't present in the URL, the node sets an empty list (`[]`) in the corresponding node state property.

- If an allowed query parameter is also listed in the **Allowed query parameters to be delimited**, the node sets the values of the listed URL parameter in the node state as a comma-delimited list.

Important

- The node URL-decodes query parameters before putting them into the node state.
  If a query parameter includes `%2C`, the node puts this value into the node state as a comma (`,`). If the query parameter is included in the **Allowed query parameters to be delimited** the node interprets `%2C` as a comma delimiter.
  The node decodes the plus symbol (`+`) as a *space*. If you need a `+` in a query parameter value, encode it as `%2B` in the URL.
- Values stored in the node state can override values in the authentication journey.
  Take special care when you configure this node so that you don't unintentionally override parameters such as usernames and passwords.
  The output of this node isn't under the control of the node itself. Encode sensitive values appropriately, either at node output, or before the values are used later in the journey.

Outcomes

Single outcome that passes an updated node state to the next node in the journey.

Errors

- `No parameters configured - this node is redundant`

  The node logs this error if you include it in a journey but don't configure any **Allowed query parameters**.

- `Cannot delimit parameter if it is not configured as a parameter to be stored in node state`

  The node logs this error if you add a parameter to the list of **Allowed query parameters to be delimited** but not to the list of **Allowed query parameters**.

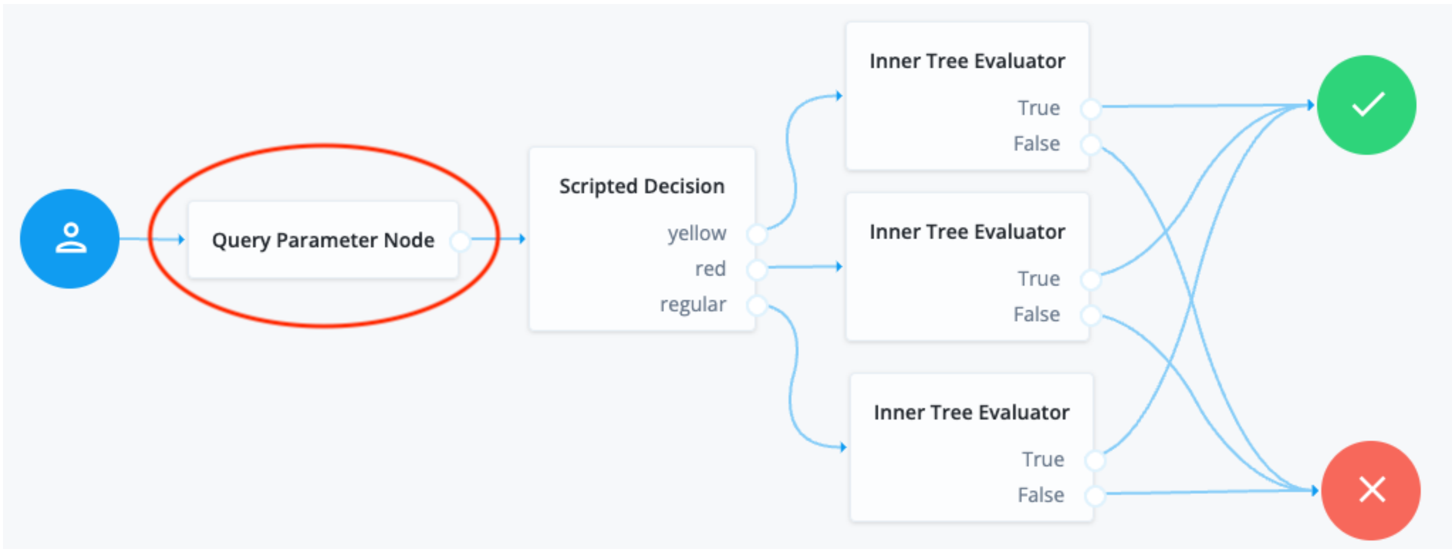
Examples

Use the Query Parameter node to customize a journey based on query parameters in the URL. The following scenarios illustrate how this node might be used:

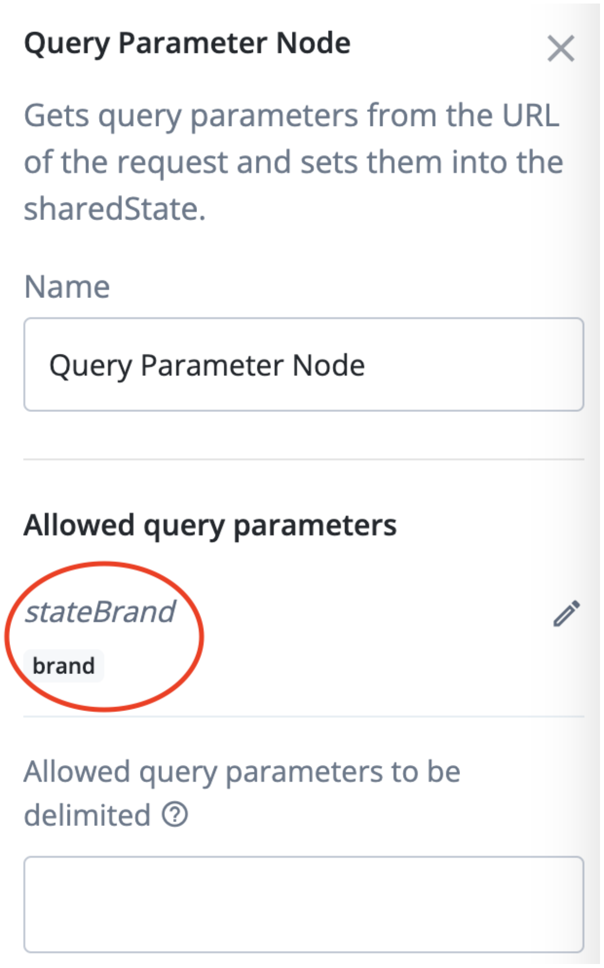
Customized branding

An organization has several *brands* that use the same journey. Use this node to customize the brand the user sees, based on the query parameters.

Consider the following authentication journey:



1. The configuration of the Query Parameter node maps the `brand` query parameter to a property in the node state named `stateBrand`



Query Parameter Node

Gets query parameters from the URL of the request and sets them into the sharedState.

Name: Query Parameter Node

Allowed query parameters

stateBrand
brand

Allowed query parameters to be delimited

2. The user accesses the journey at a URL that can be one of the following:

   ◦ `https://<tenant-env-fqdn>/am/XUI/?realm=alpha&authIndexType=service&authIndexValue=Login`

   - `https://<tenant-env-fqdn>/am/XUI/?realm=alpha&authIndexType=service&authIndexValue=Login&brand=yellow`
   - `https://<tenant-env-fqdn>/am/XUI/?realm=alpha&authIndexType=service&authIndexValue=Login&brand=red`

3. The Query Parameter node obtains the value of `brand` from the URL and sets that value in the `stateBrand` property in the node state; for example, `stateBrand=yellow`

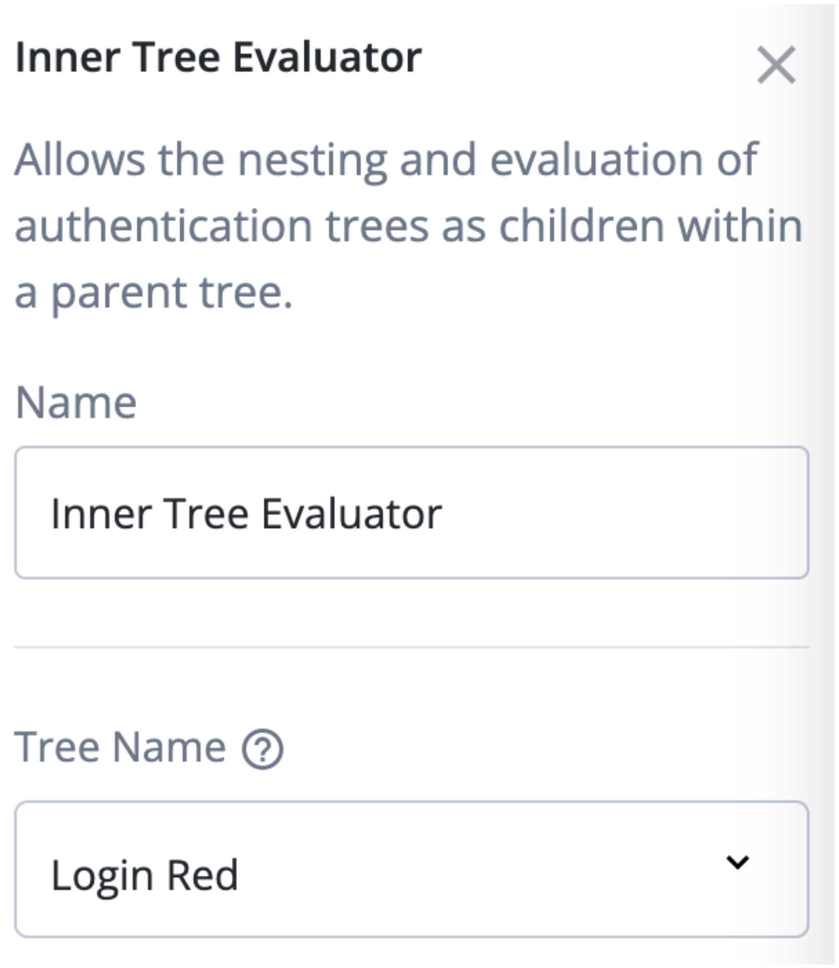
4. The journey progresses to the scripted decision node that includes the following script:

```
var brand = JSON.parse(nodeState.get('stateBrand'));

if (brand.indexOf("yellow") >= 0) {
  outcome = "yellow";
} else if (brand.indexOf("red") >= 0) {
  outcome = "red";
} else {
  outcome = "regular";
}
```

5. The script routes the journey to one of three outcomes; `yellow`, `red`, or `regular`, depending on the value of the `stateBrand` property.

6. The outcomes direct the user to a custom branded Login journey configured in an Inner Tree Evaluator node; for example:



Inner Tree Evaluator

Allows the nesting and evaluation of authentication trees as children within a parent tree.

Name: Inner Tree Evaluator

Tree Name: Login Red

7. Each Inner Tree Evaluator node routes the end user to a login journey that uses a custom brand.

Redirection from an external system

An external system redirects a user to this authentication journey. The external system must share information about the user with the journey. Use this node to obtain the relevant query parameters and inform the journey of their values.

Register Logout Webhook node

Registers the specified webhook to trigger when a user's session ends. The webhook triggers when a user explicitly logs out or the maximum idle time or expiry time of the session is reached.

The webhook is only registered if evaluation passes through this node. You can register multiple webhooks during the authentication process, but they must be unique.

Find more information on webhooks in Configure authentication webhooks.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

Single outcome path.

Properties

Property	Usage
Webhook name	Specify the name of the webhook to register.

Remove Session Properties node

Removes properties from the session. The session properties may have been set by a Set Session Properties node elsewhere in the flow.

If a specified key is not found in the list of session properties it is added to the session upon successful authentication, no error is thrown, and evaluation continues along the single outcome path.

If a specified key is found, the evaluation continues along the single outcome path after setting the value of the property to `null`.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

Single outcome path.

Properties

Property	Usage
Property Names *(required)*	Enter one or more key names of properties to remove from the session.

Request Header node

The **Request Header** node lets you insert values from request headers into configurable node state properties. This lets you customize journeys based on the request header values.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node takes its inputs from one or more request headers and maps these values to properties in the node state. For multi-valued headers, the node delimits headers with a comma (`,`) and assigns the value to the node state property as a comma-delimited list.

The node doesn't read input from the state.

The node has no required predecessor nodes.

Dependencies

None.

Configuration

Property	Usage
Allowed headers	List the header names the node can obtain from the URL and map them to a property in the node state: 1. Click **Add**. 2. In the **Key** field, enter the query parameter name. 3. In the **Value** field, enter the node state property that will hold the value of this request header. This field sets an *allowlist* of headers the node can pull into the node state. Exercise caution when you create this list to avoid injecting harmful data into the node state. If a provided request header isn't in this list, the node ignores it. To edit an entry, click the **Pencil** icon (✎). To remove an entry, click the **Delete** icon (🗑).
Allowed headers to be delimited	The allowed headers that can take multiple values and whose values you want to store in the node state in a comma-delimited list; for example, `["yellow", "green", "red"]`. Enter the header name in the **Add value** field and click **Add**. If you don't delimit the values of a multi-valued header, the node stores the values in the node state as a single string value; for example, `["yellow, green, red"]` . To edit an entry, click the **Pencil** icon (✎). To remove an entry, click the **Delete** icon (🗑).

Outputs

- If the **Allowed headers** setting has one or more values, the node adds the values of the listed request headers to the corresponding properties in the node state.
- If the **Allowed headers to be delimited** setting has a value but that header isn't provided in the request, the node sets an empty list (`[]`) in the corresponding node state property.
- If an allowed header is also listed in the **Allowed headers to be delimited**, the node sets the values of that header in the node state as a comma-delimited list.
- The node performs no decoding or sanitization on the header value. It simply passes the value into the node state as a string.

Important

Values stored in the node state can override values in the authentication journey.
Take special care when you configure this node so that you don't unintentionally override parameters such as usernames and passwords.
The output of this node isn't under the control of the node itself. Encode sensitive values appropriately, either at node output, or before the values are used later in the journey.

Outcomes

Single outcome that passes an updated node state to the next node in the journey.

Errors

- `No headers configured – this node is redundant`

  The node logs this error if you include it in a journey but don't configure any **Allowed headers**.

- `Cannot delimit header if it is not configured as a header to be stored in node state`

  The node logs this error if you add a header to the list of **Allowed headers to be delimited** but not to the list of **Allowed headers**.

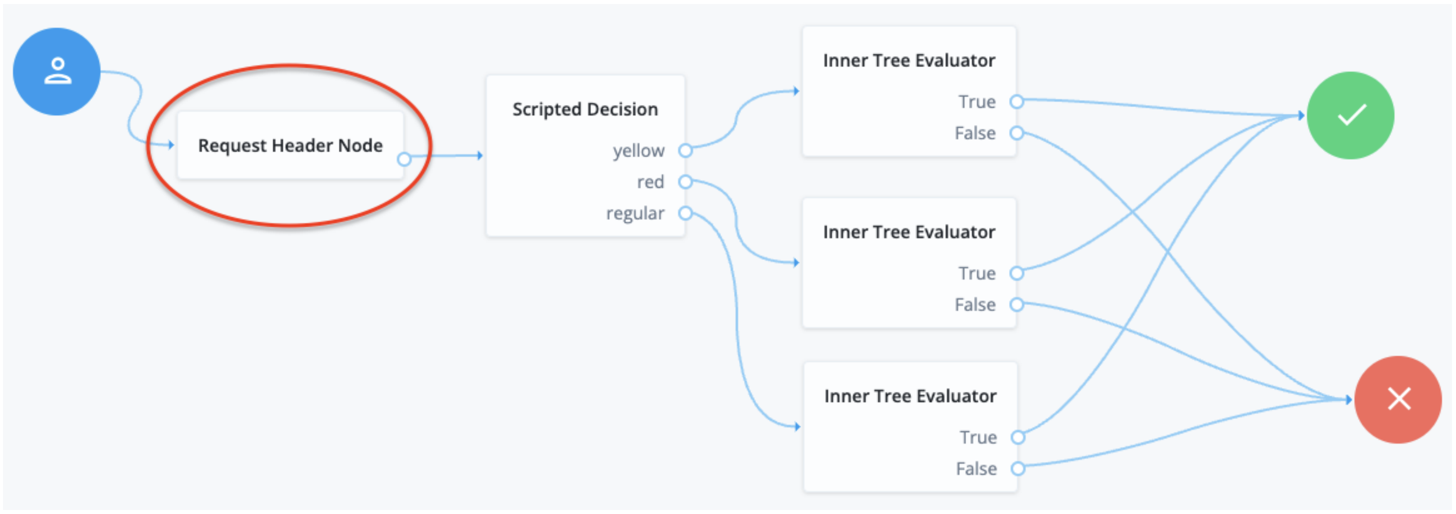
Examples

Use the Request Header node to customize a journey based on the values of specific request headers. The following scenarios illustrate how this node might be used:

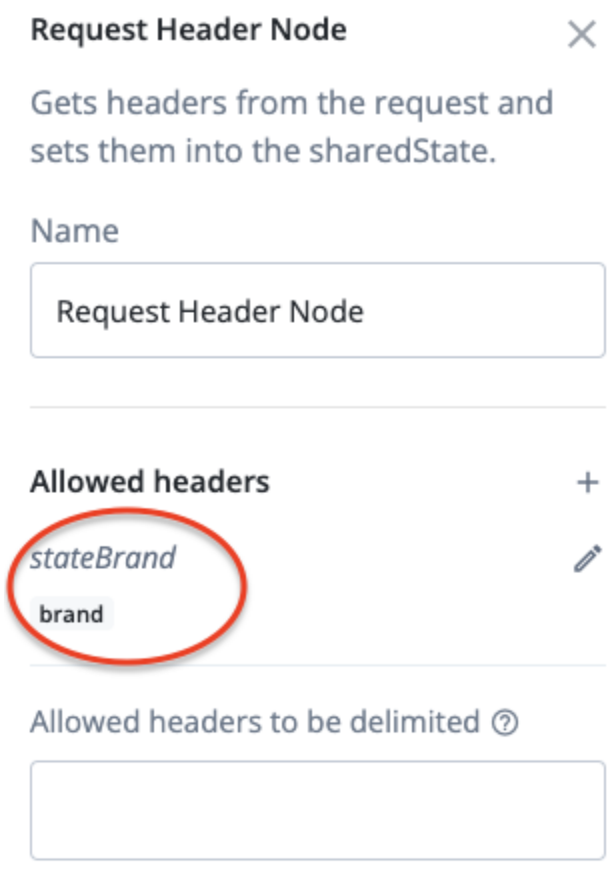
Customized branding

An organization has several *brands* that use the same journey. Use this node to customize the brand the user sees, based on the request header.

Consider the following authentication journey:



1. The configuration of the Request Header node maps the `brand` header to a property in the node state named `stateBrand`



Gets headers from the request and sets them into the sharedState.

Name: Request Header Node

Allowed headers

stateBrand

brand

Allowed headers to be delimited

2. The REST request to access the journey includes one of the following headers:

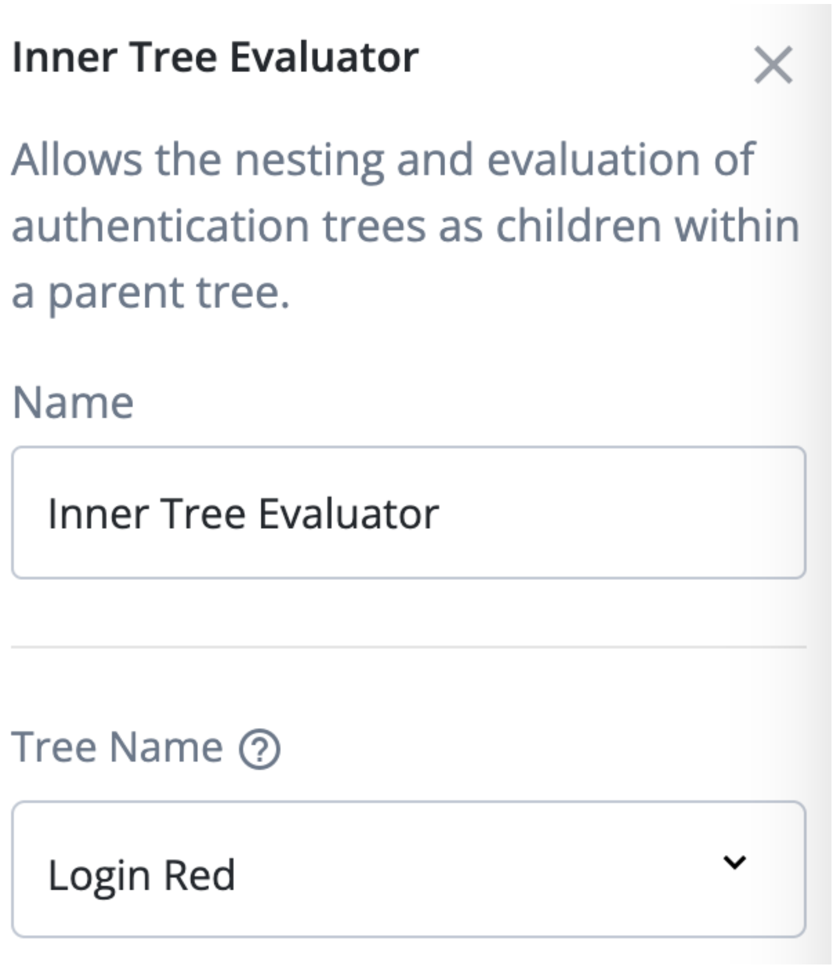
   ◦ `--header "brand: yellow"`
   ◦ `--header "brand: red"`
   ◦ `--header "brand: regular"`

3. The Request Header node obtains the value of the `brand` header and sets that value in the `stateBrand` property in the node state; for example, `stateBrand=yellow`
4. The journey progresses to the scripted decision node that includes the following script:

```
var brand = JSON.parse(nodeState.get('stateBrand'));

if (brand.indexOf("yellow") >= 0) {
  outcome = "yellow";
} else if (brand.indexOf("red") >= 0) {
  outcome = "red";
} else {
  outcome = "regular";
}
```

5. The script routes the journey to one of three outcomes; `yellow`, `red`, or `regular`, depending on the value of the `stateBrand` property.
6. The outcomes direct the user to a custom branded Login journey configured in an Inner Tree Evaluator node; for example:



Inner Tree Evaluator

Allows the nesting and evaluation of authentication trees as children within a parent tree.

Name: Inner Tree Evaluator

Tree Name: Login Red

7. Each Inner Tree Evaluator node routes the end user to a login journey that uses a custom brand.

Redirection from an external system

An external system redirects a user to this authentication journey. The external system must share information about the user with the journey. Use the Request Header node to obtain the relevant request headers and inform the journey of their values.

Retry Limit Decision node

The **Retry Limit Decision** node tracks failed authentications. If the number of failed authentications is below a specified **Retry Limit**, the user can attempt authentication again. Otherwise, the node continues evaluation along the `Reject` outcome path.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

The node reads the `username` from the shared state. Implement the following node before this node in the journey:

- Username Collector node (standalone AM)
- Platform Username node (Ping Identity Platform deployment)

The node also reads the `realm` from the shared state.

Dependencies

This node has no dependencies.

Configuration

Property	Usage
Retry limit	Specify the number of retries to allow. Default: `3`

Property	Usage
Save Retry Limit to User	Specify whether the number of failed login attempts persists across multiple journeys until authentication is successful. Possible values are: ***Enabled*** The node saves the number of failed login attempts to the `retryLimitNodeCount` attribute in the user's profile. New flows using this node start with the stored value and continue to the retry limit. AM resets the count after the user authenticates successfully with an authentication journey that contains this node. If AM can't find the user's profile, authentication ends with an error. ***Disabled*** The node saves the number of failed login attempts in a shared state property named `nodeId.retryCount` and discards the value when the authentication journey ends. For security reasons, you should enable this setting. Default: Enabled.

Outputs

If **Save Retry Limit to User** is enabled, the node increments the retry count and saves the number of failed attempts to the `retryLimitNodeCount` attribute in the user's profile. If the user can't be identified during the journey, the journey ends with an error.

If **Save Retry Limit to User** is disabled, the node increments the retry count and saves the number of failed attempts to a shared state property named `nodeId.retryCount`. The count is lost if the journey is restarted.

Outcomes

`Retry`

The user hasn't exceeded the number of allowed retries and can attempt authentication again.

`Reject`

The user has exceeded the number of allowed retries.

Errors

This node can log the following:

Warnings

- `Error clearing attribute`

  The node can't reset the `retryLimitNodeCount` user attribute after the user has successfully authenticated.

Errors

- `Error getting current retry count`

  The node can’t retrieve the current retry count.

- `Failed to save retryLimitNodeCount to user: Identity Repo has not been upgraded.`

  The node can’t save retry count details to the `retryLimitNodeCount` user attribute during the authentication flow.

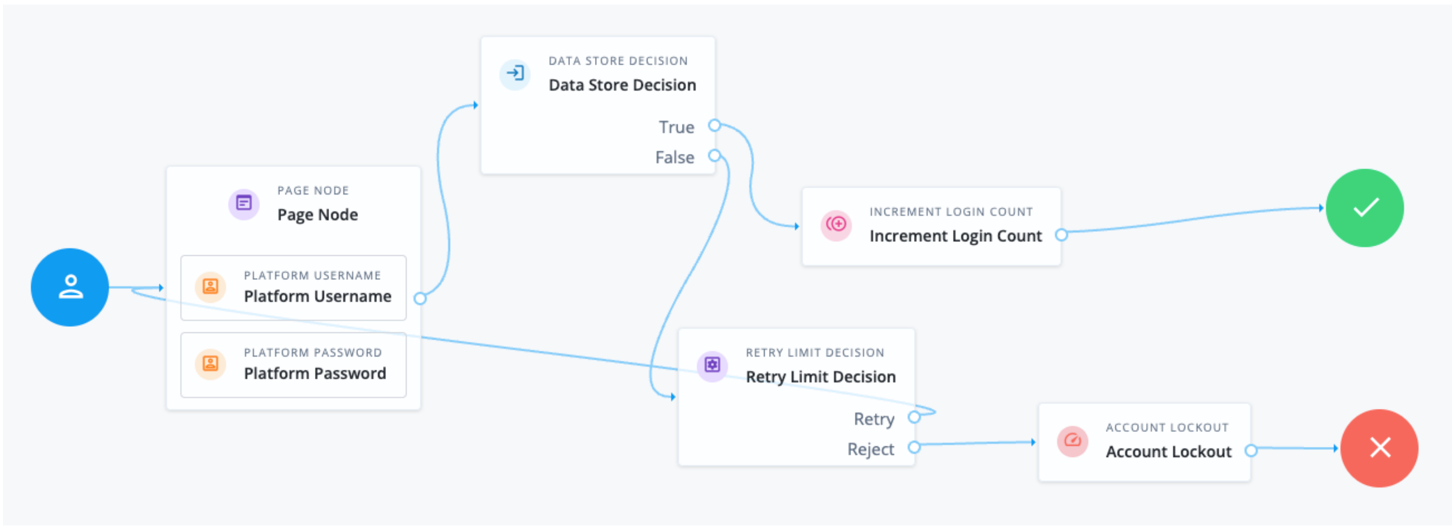
- `Error setting retry count on user attribute`

  The node can’t increment the retry count on the `retryLimitNodeCount` user attribute during the authentication flow.

These warnings and errors typically occur if the identity store is unreachable or the user no longer exists.

Examples

This example uses the **Retry Limit Decision** node to allow a user three attempts to authenticate. Otherwise, their account is locked.



- The Page node containing the Platform Username node and Platform Password node prompts for credentials.
- The Data Store Decision node validates the username-password credentials.
- The Increment Login Count node updates the number of successful authentications in the user profile.
- The **Retry Limit Decision** node is configured to allow three login attempts and either retries the login attempt or rejects it depending on the number of failed attempts.
- The Account Lockout node locks the user’s account on their fourth failed attempt.

Scripted Decision node

The **Scripted Decision** node lets you run a server-side script in an authentication journey. It exists to let you connect the script to other nodes with the journey editor.

The script makes a decision to set the outcome for the node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

Scripted Decision node inputs depend entirely on the node's server-side script.

The script has access to the authentication context including:

- Headers from the request
- Query string parameters from the request
- Secrets configured for the realm
- Shared state data
- User profile data

The script can use callbacks to prompt the user for information.

Find details about the inputs available to the script in Scripted decision node API.

You can restrict available inputs using the **Script Inputs** field when configuring the node.

Dependencies

A **Scripted Decision** node depends on a **Decision node for authentication trees**] script you create before you configure the node.

Configuration

Property	Usage
Script	Select the script to run from the drop-down list.
Outcomes	Enter one string for each `outcome` the script can set. The node shows only the outcomes you configure. If you omit an `outcome` string, you can't connect it in the journey editor. When the script sets an `outcome` you omitted in the configuration, it logs a warning. This can prevent the journey from completing successfully.

Property	Usage
Script Inputs	Optionally, list the shared state data properties required by the script. If you change the setting, you must declare each property or `null` for no properties. Default: `*`. The script has access to all shared and transient state data. **Important** Sensitive data in transient state upgrades to *secure* state if: • The node sends a callback to the user. • The node detects a downstream node requesting the transient state data as input. Unless the downstream node explicitly requests the secure state data by name, the authentication journey removes it from the node state after processing the next callback. For example, a node in a registration journey stores a user's password in transient state. The node sends a callback to the user before an inner tree node, downstream in the journey, consumes that password. As part of the callback, the journey assesses what to add to the secure state. It does this by checking the state inputs that downstream nodes in the journey require. Nodes that *only* request `*` are ignored, as this would result in putting everything in transient state into secure state, and retaining sensitive information longer than necessary. If a downstream node requires the password, it must explicitly request it as state input, even if it lists the `*` wildcard as input.
Script Outputs	Optionally, list the shared state data properties the node expects the script to set. If you change the setting, you must declare each property or `null` for no properties. Default: `*`. The node doesn't validate the script outputs at all.

Outputs

Scripted Decision node outputs, such as updates to shared state data, depend entirely on the node's server-side script.

You can restrict available outputs using the **Script Outputs** field when configuring the node.

Outcomes

The script defines the outcomes by setting its `outcome` variable to an outcome string before returning.

You include all possible `outcome` strings from the script in the **Outcome** field when configuring the node.

The authentication journey continues along the outcome path from the script.

Errors

The server-side script can log messages.

The node logs the following warning messages:

Warnings:

- `Found an action result from scripted node, but it was not an Action object`: An action in a legacy script didn't return an object with type `Action`.
- `Found an action result from scripted node, but it was not an ActionWrapper object`: An action in a next generation script didn't return an object with type `ActionWrapper`.
- `invalid script outcome <outcome>`: The <outcome> is missing in the **Outcome** field of the node configuration.
- `invalid script outcome <action-outcome> in action`: The <action-outcome> is missing in the **Outcome** field of the node configuration.
- `script outcome error`: The script set an outcome not found in the **Outcome** field of the node configuration.

Examples

You use a **Scripted Decision** node when no other available node does what you need.

In this example, the node depends on the following JavaScript **Decision node for authentication trees**] script. The script gets the user's names from their profile and stores a message in a shared state property:

Next-generation

```
// Get the username from shared state data:
var username = nodeState.get('username')

// Get the given name(s) and surname(s) from the user profile:
var profile = idRepository.getIdentity(username)
var givenname = profile.getAttributeValues('givenName')
var surname = profile.getAttributeValues('sn')
if (!(givenname && surname)) {
  var error = `Failed to get names for ${username}: ${givenname} ${surname}`
  action.goTo('Failure').withErrorMessage(error);
} else {
  // Record who authenticated in the shared state data:
  var firstGivenName = givenname[0]
  var firstSurname = surname[0]
  var now = new Date().toLocaleString()
  var message = `${firstGivenName} ${firstSurname} logged in at ${now}.`
  nodeState.putShared('message', message)
  action.goTo('Success');
}
```

Legacy

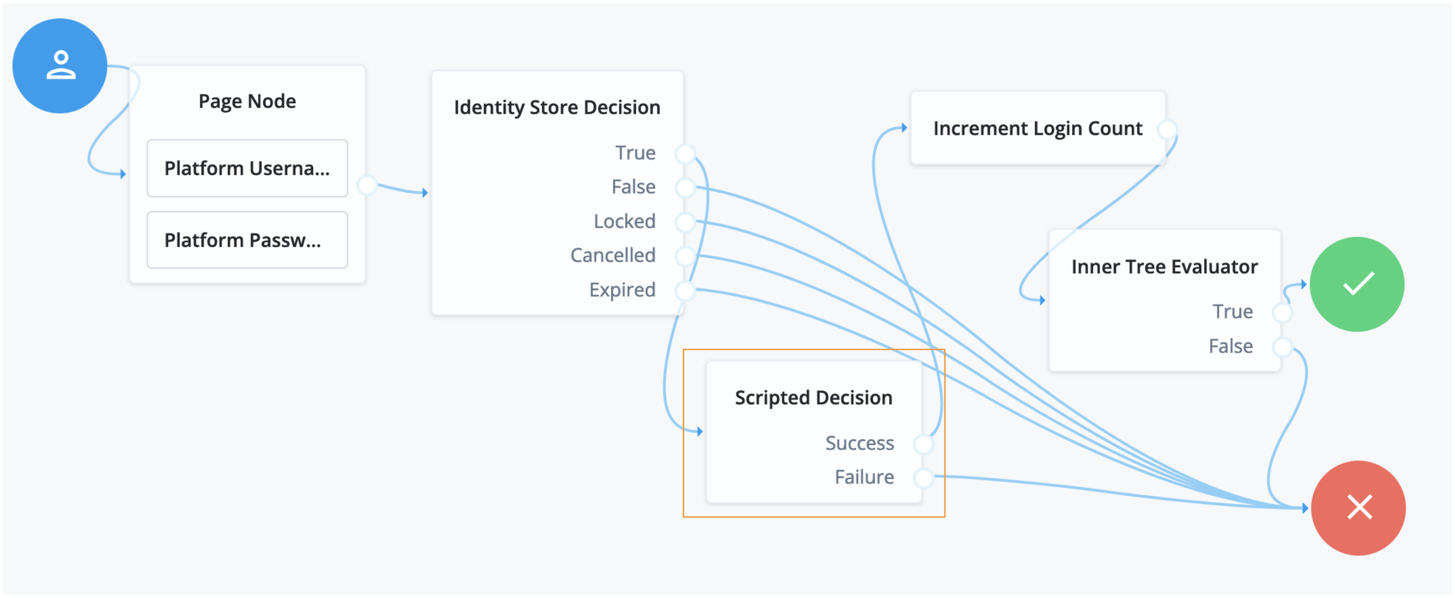
```
var goTo = org.forgerock.openam.auth.node.api.Action.goTo

// Get the username from shared state data:
var username = nodeState.get('username').asString()

// Get the given name(s) and surname(s) from the user profile:
var profile = idRepository.getIdentity(username)
var givenname = profile.getAttributeValues('givenName')
var surname = profile.getAttributeValues('sn')
if (!(givenname && surname)) {
  var error = `Failed to get names for ${username}: ${givenname} ${surname}`
  action = goTo('Failure').withErrorMessage(error).build()
} else {
  // Record who authenticated in the shared state data:
  var firstGivenName = givenname[0]
  var firstSurname = surname[0]
  var now = new Date().toLocaleString()
  var message = `${firstGivenName} ${firstSurname} logged in at ${now}.`
  nodeState.putShared('message', message)
  action = goTo('Success').build()
}
```

Notice the script sets the outcomes using the `Action.goTo(outcome)` function.

The journey is as follows:



1. The Page node prompts the user for their username and password.

2. Replace the Identity Store Decision node with a Data Store Decision node to check the username and password.

3. The **Scripted Decision** node runs the script and has the following settings:

Script	The name of the script
Outcomes	`Success`, `Failure`
Script Inputs	`username`
Script Outputs	`*`

Notice the **Outcomes** setting lists all outcome strings from the script.

4. The Increment Login Count node updates the count on successful authentication.

5. The Inner Tree Evaluator node refers to another journey to perform more steps.

   This node is optional.

If you activate debug mode for the journey and select **Enable Debug Popup**, you find the message in the debug popup window when authenticating:

```
{
  "universalId": "id=<_id>,ou=user,o=alpha,ou=services,ou=am-config",
  "transactionId": "<transaction-id>",
  "password": "<password>",
  "pageNodeCallbacks": {
    "0": 0,
    "1": 1
  },
  "realm": "/alpha",
  "message": "Babs Jensen logged in at August 16, 2023 9:55:33 AM UTC.",
  "authLevel": 0,
  "objectAttributes": {
    "password": "<password>"
  },
  "username": "id=<_id>"
}
```

Set Session Properties node

The **Set Session Properties** node adds `key:value` properties to the user's session on successful authentication.

Tip

You can access session properties using a variable in a webhook. Find more information in Configure authentication webhooks.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes

Product	Available?
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

None. This node does not read shared node state data.

Dependencies

Ensure the user can successfully authenticate and get a session.

If the user never gets a session, this node has no effect.

Configuration

Property	Usage
Properties	The session properties to set. • To add a session property: 1. Click **+**, then **+ Add** in the **Properties** modal. 2. Enter the session property name in the **Key** field and the value to set in the **Value** field. 3. Click **Done**. • To edit a property: 1. Click its pencil icon (✎). 2. Update the **Key** and **Value** as when adding properties. • To remove a property, click its delete icon (🗑). When finished, click **Save** to keep your settings.

Outputs

This node sets *session* properties; it does not change the shared state data.

This node cannot override system session properties, such as the principal or the authentication level. Use a different journey to re-authenticate the user rather than trying to change such properties with this node.

Outcomes

Single outcome path: when the journey completes successfully, this node sets the configured session properties.

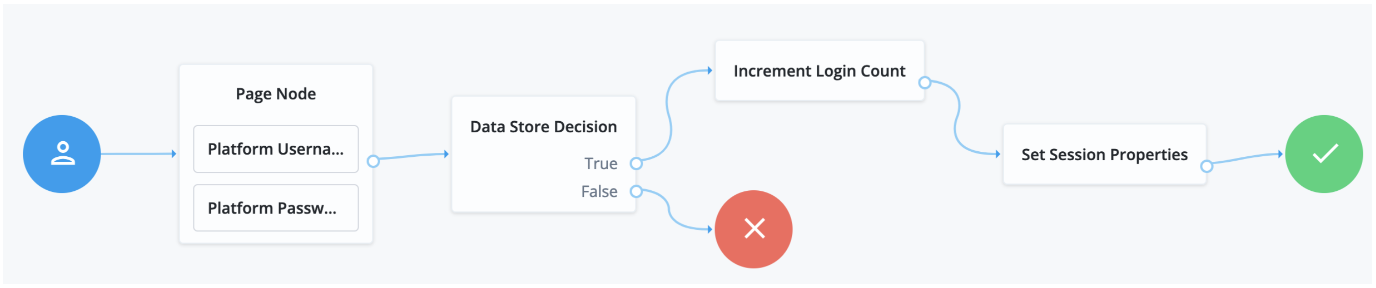
Errors

This node does not log messages of its own.

Examples

The following example uses the **Set Session Properties** node to update the `successURL` session property.

- A first platform journey updates the session property on successful authentication:

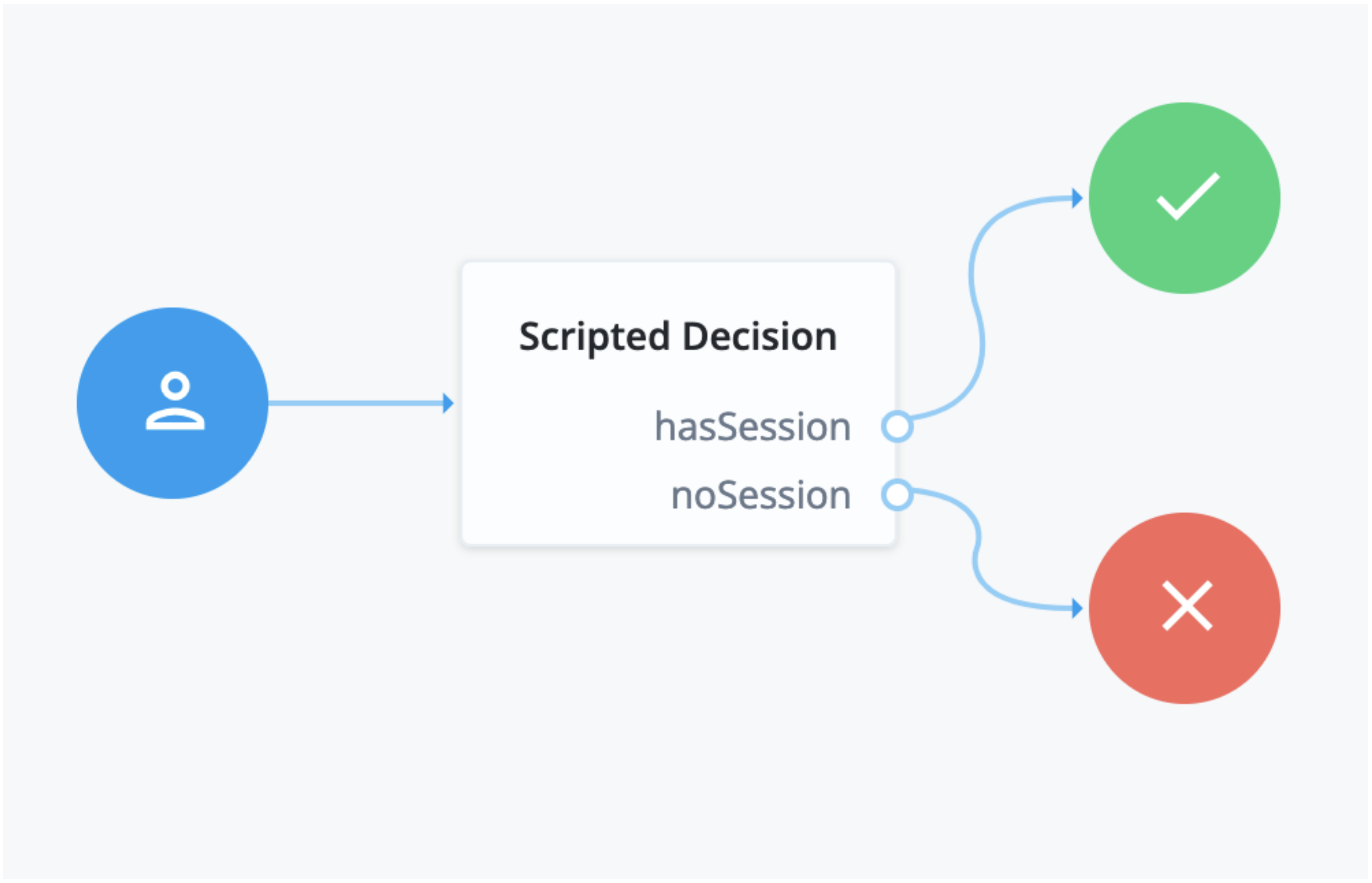
  

  - The Page node containing the Platform Username node and Platform Password node prompts for credentials.
  - The Data Store Decision node validates the username-password credentials.
  - The Increment Login Count node updates the number of successful authentications in the user profile.
  - The **Set Session Properties** node, sets the `successURL` session property.

    Configure the **Properties** to add a `successURL` property with the URL of your choice.

  When the journey completes successfully, AM updates the `successURL` in the user's session data.

- A second journey uses a script to display the session properties after the user authenticates:



The Scripted Decision node calls the following script to inject the session properties into the shared state data so the journey can display them though a debug popup:

```
if (typeof existingSession !== 'undefined') {
  nodeState.putShared('session', existingSession)
  action.goTo('hasSession')
} else {
  nodeState.putShared('session', null)
  action.goTo('noSession')
}
```

The second journey has **Debug mode** and **Enable Debug Popup** selected.

Follow these steps to try the example:

1. Create both journeys using the journey editor.
2. Sign in through the first journey with a test user account.

   The browser shows the user profile page.
3. In the same browser window, browse to the URL for the second journey.

   The debug popup window displays the shared state data including session properties:

```
{
    "transactionId": "...",
    "session": {
        "successURL": "<your-success-url>",
        "...": "..."
    },
    "realm": "/alpha",
    "authLevel": 0,
    "username": "test"
}
```

The `successURL` property is set to <your-success-url>, the one you configured as the value in **Properties** of the **Set Session Properties** node.

4. Sign out as the test user.
5. Sign in through the *default* journey as the test user.

   The default journey doesn't use the **Set Session Properties** node with your configuration, so it uses the default value for the `successURL` session property.
6. In the same browser window, browse to the URL for the second journey again.

   The debug popup window displays the shared state data, including session properties with the default `successURL` value.

State Metadata node

The **State Metadata** node returns selected attributes from the shared node state as metadata.

This node sends a `MetadataCallback` to retrieve shared state values, which it adds to the JSON response from the `/authenticate` endpoint. This example shows how a shared state attribute, `mail`, is returned:

```
{
    "callbacks": [
        {
            "type": "MetadataCallback",
            "output": [
                {
                    "name": "data",
                    "value": {
                        "mail": "bjensen@example.com"
                    }
                }
            ]
        }
    ]
}
```

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

This node reads its configured **Attributes** from the shared node state.

Dependencies

None.

Configuration

Property	Usage
Attributes	Specify one or more shared state attribute names for return. Default: none

Outputs

This node only sends the callback. It does not modify the shared node state.

Outcomes

Single outcome path.

Evaluation continues after the callback.

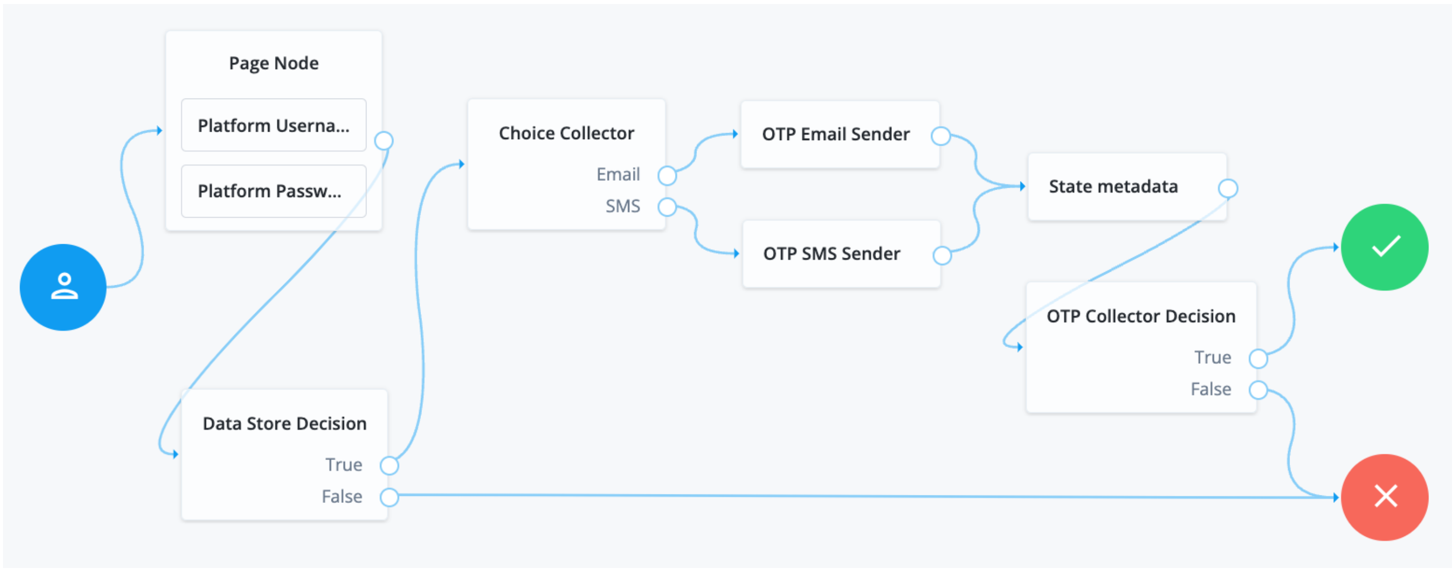
Errors

This node does not log error or warning messages of its own.

Example

Use this node to display custom information that includes user attributes without having to alter the existing flow.

For example, for OTP authentication with a choice of email or SMS, use this node to return the user's email address or phone number. You can use the attributes with an OTP Collector Decision node, and optionally, a Scripted Decision node, to customize the data for display later.



- The Page node with a Platform Username node and Attribute Collector node prompts for the credentials.
- The Data Store Decision node confirms the user's credentials.
- The Choice Collector node lets the user opt for notification through email or a text message.
- The OTP Email Sender node sends the one-time password (OTP) as email.
- The OTP SMS Sender node sends the OTP as a text message.
- The **State Metadata** node injects attributes for additional information.
- The OTP Collector Decision node displays the additional information when collecting the OTP to verify.

Success URL node

Sets the redirect URL when authentication succeeds.

Note

Specifying a success URL overrides any `goto` query string parameters.

Find information on how AM determines the redirection URL, and on configuring the Validation Service to trust redirection URLs, in Configure success and failure redirection URLs.

Tip

The URL is also saved in the `nodeState` object on the `successUrl` key.
Learn more in Customize authentication trees.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Properties

Property	Usage
Success URL *(required)*	Specify the full URL to redirect to when the authentication succeeds.

Timer Start node

Starts a named timer metric, which you can stop with a Timer Stop node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

Single outcome path.

Properties

Property	Usage
Start Time Property	Specify a property name into which to store the current time. Specify the same value in any instances of the Timer Stop node that measure the time elapsed since evaluation passed through this node.

Timer Stop node

Records the time elapsed since evaluation passed through the Timer Start node in the specified metric name.

For information on the `Timer` metric type, refer to Monitoring metric types.

Note that this node does not reset the time stored in the specified **Start Time Property** property. Other Timer Stop nodes can also calculate the time elapsed since evaluation passed through the same Timer Start node.

The metric is exposed in all available interfaces, as described in Monitor AM instances.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

Single outcome path.

Properties

Property	Usage
Start Time Property	Specify the property name containing the time from which to calculate the elapsed time.
Metric Key *(required)*	Enter the name for a new metric that stores the calculated elapsed time. The name that you select is used to identify the metric that exposes the data collected by this node. For example, if you enter `calculated.time`, AM exposes a new metric with this name to the Common REST, JMX, or Graphite interfaces. If you use Prometheus, the name is prefixed with `am_` and appended with `_seconds` to become `am_calculated_time_seconds`.] **Tip** Metrics collate data from multiple invocations of a journey. To record the time it takes for a particular journey to complete, use a Scripted Decision node to store the start time in shared state. Use a script at the end of the journey to capture the end time and output the calculated journey time to the authentication audit logs. Find more information in Audit information.

Thing nodes

Authenticate Thing node

This node authenticates a *thing*. A thing represents an IoT device, service, or the IoT Gateway.

Before you configure this node, configure the IoT Service for the realm.

Important

Support for this node is provided by the IoT SDK.

The node supports two methods of authentication:

1. Proof of Possession JWT

   The node collects a proof-of-possession JWT from the request and does the following:

   - Checks that the claims are valid.
   - Checks that an identity with the same ID as the name of the JWT subject exists.
   - Checks that the identity contains a confirmation key that matches the JWT `kid`.
   - Validates the JWT signature, using the confirmation key stored in the identity.

2. Client Assertion

   The node collects a JWT Bearer token from the request for authentication and validates the request according to the JWT Profile for OAuth 2.0 Client Authentication and Authorization Grants.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `Success`
- `Failure`

- `Requires Registration`

If all checks are successful, evaluation continues through the `Success` path, and adds the username and the verified claims to the shared node state.

If the identity does not exist, or AM cannot match the identity with the confirmation key, evaluation continues through the `Requires Registration` outcome.

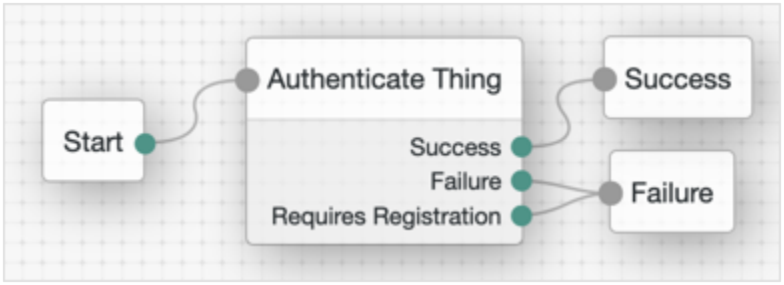
If any other check fails, evaluation continues through the `Failure` outcome.

Properties

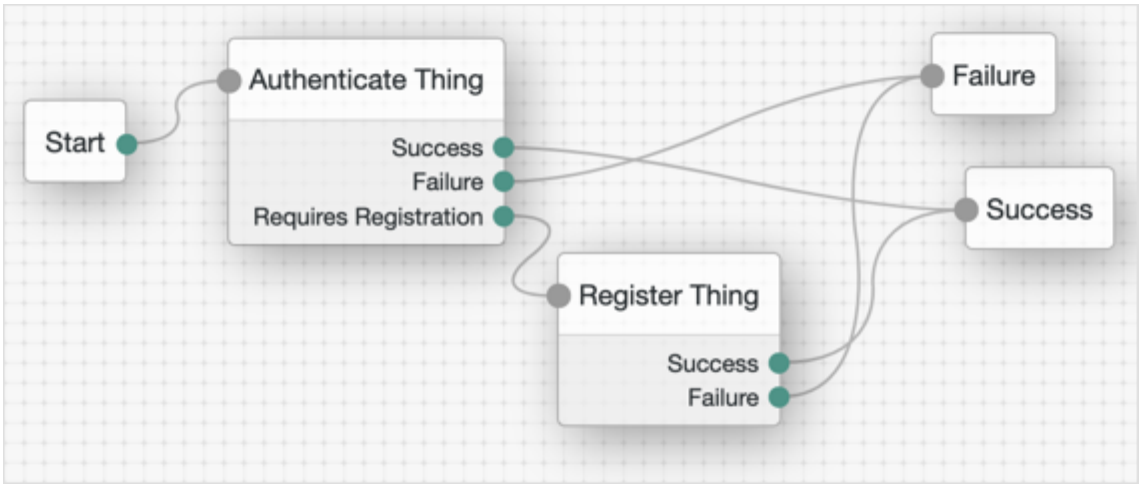
Property	Usage
JWT Authentication Method	Choose the required JWT authentication method: ***Proof of Possession*** Prove that the signer of the JWT is the owner of the key by including a challenge nonce in the JWT. Validation is according to the JWT Proof of Possession specification. ***Client Assertion*** Present a JWT Bearer token for authentication and validate the request according to the JWT Profile for OAuth 2.0 Client Authentication and Authorization Grants.
Issue Restricted Token	If this setting is enabled, the node adds a Proof of Possession restriction to the session token issued on successful authentication. Any requests accompanied by the token must be signed with the key that was used to sign the authentication JWT.
Additional Audience Values	Specify any additional audience values that will be permitted when verifying JWTs. These audience values are in addition to the AM base, issuer and token endpoint URIs for the Client Assertion authentication method or the realm path for Proof of Possession.

Examples

The following example shows how to authenticate a thing when the identity already exists in the identity store and when its profile contains a confirmation key:



The following example shows how to authenticate a thing when the identity does not exist, or when it needs to refresh its confirmation key:



Register Thing node

This node authenticates a *thing*. A thing represents an IoT device, service, or the IoT Gateway.

Before you configure this node, configure the IoT Service for the realm.

Important

Support for this node is provided by the IoT SDK.

The node collects a JWT from the request and validates the JWT according to the configured JWT registration method.

If the JWT is valid, the node uses the claims in the JWT to create an identity for the thing and register (or rotate) a confirmation key for it. Then, evaluation continues through the `Success` outcome.

If the node cannot validate the JWT, evaluation continues through the `Failure` outcome.

For an example on how to use this node, refer to Authenticate Thing node.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Outcomes

- `Success`

- `Failure`

Properties

Property	Usage
JWT Registration Method	Choose the method to validate the JWT: ***Proof of Possession & Certificate*** Register using a Proof of Possession JWT that includes an X.509 certificate for providing trust. A challenge nonce is presented in the callback and must be included in the signed JWT. ***Proof of Possession & Software Statement*** Register using a Proof of Possession JWT and a Software Statement for providing trust. A challenge nonce is presented in the callback and must be included in the signed Proof of Possession JWT. The claims in the Software Statement take precedence over the claims in the Proof of Possession JWT. ***Proof of Possession*** Register using a Proof of Possession JWT without using a trusted third party. A challenge nonce is presented in the callback and must be included in the signed JWT. ***Software Statement*** Register using a Software Statement, without doing proof of possession. If you select this registration method, the resultant session token will not include a proof of possession restriction. Default: Proof of Possession & Certificate
Verify Certificate Subject	If the configured JWT registration method is `Proof of Possession & Certificate`, this option verifies that the subject provided in the JWT is the same as the X.509 certificate subject CN or UID. Default: Enabled
Create Identity	Specifies whether AM will create an ID for the thing if one does not exist. Default: Disabled
Rotate Confirmation Key	Specifies whether multiple confirmation keys can be registered for a thing. Disable this setting to allow only one key per thing. Default: Disabled
Default Attribute Values	Lets you set default values for the thing's attributes, where **KEY** is the name of the attribute in the data store, and **VALUE** is the default value of the attribute.
Claim to Attribute Mapping	If **Create Identity** is enabled, this property lets you map verified claims in the JWT to attributes in the thing identity. **KEY** is the claim name and **VALUE** is the name of the attribute in the data store.

Property	Usage
Overwrite Attributes	Specifies whether the node overwrites the value for an existing profile attribute when a claim with a different value is provided in the JWT. Default: Disabled

Uncategorized nodes

Debug node

Displays debug information about the current authentication tree.

This node collects information, such as the shared node state, the identity object's `universalId`, and the transaction ID, which are useful for reference in log messages.

Availability

Product	Available?
PingOne Advanced Identity Cloud	No
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	No

Outcomes

Single outcome path.

Properties

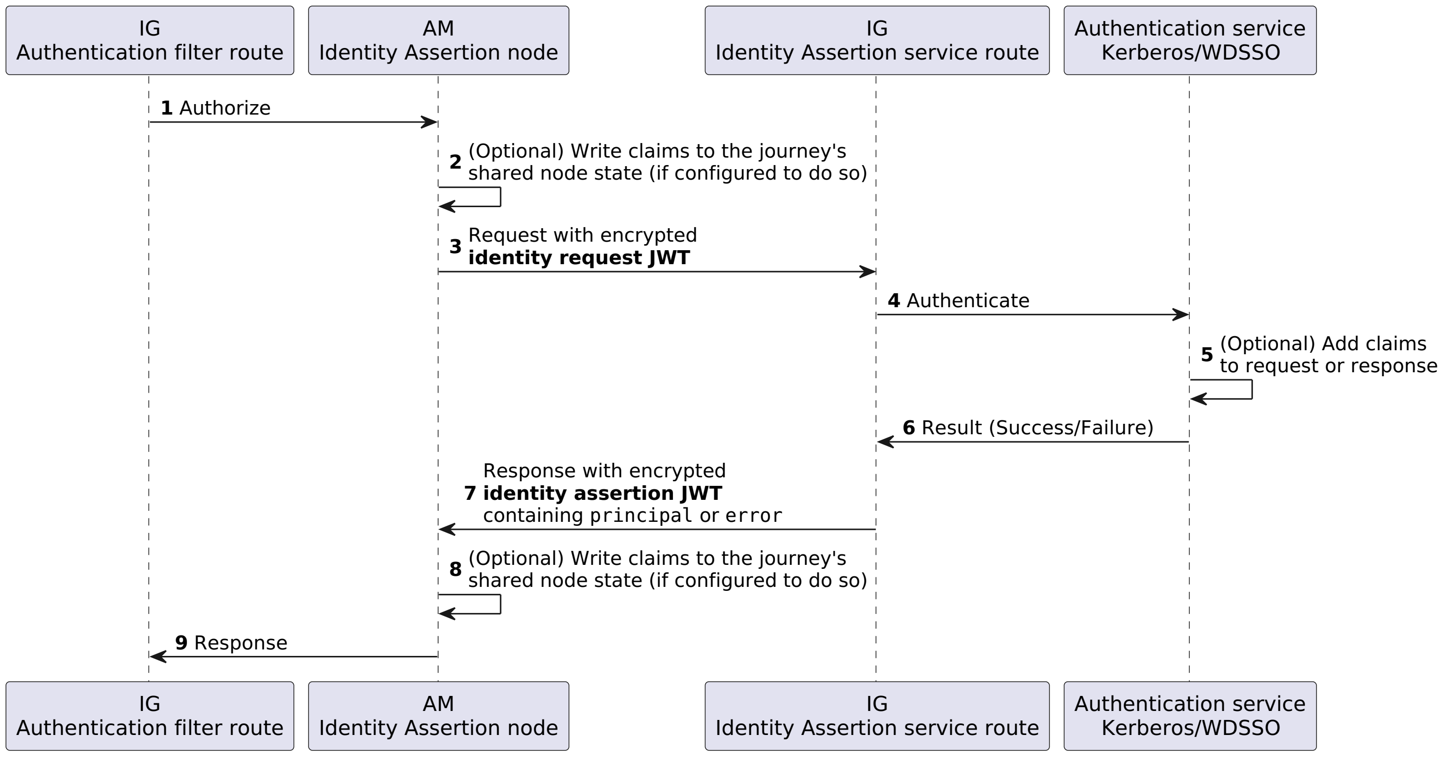
Property	Usage
Enable Debug Popup	If enabled, a popup window displays debug logs as you step through the flow in a browser.

Identity Assertion node

The **Identity Assertion** node provides a secure communication channel for authentication journeys to communicate directly with PingGateway.

The node extends AM by adding PingGateway's routing capabilities and supporting identity assertion with third-party authentication services. Authentication services include Windows Desktop SSO and Kerberos.

The following image shows the flow of an authentication request:


IG
Authentication filter route
AM
Identity Assertion node
IG
Identity Assertion service route
Authentication service
Kerberos/WDSSO
1 Authorize
2 (Optional) Write claims to the journey's shared node state (if configured to do so)
3 Request with encrypted identity request JWT
4 Authenticate
5 (Optional) Add claims to request or response
6 Result (Success/Failure)
7 Response with encrypted identity assertion JWT containing principal or error
8 (Optional) Write claims to the journey's shared node state (if configured to do so)
9 Response

AM and PingGateway share a symmetric key for encryption and decryption at both ends of the flow.

Availability

Product	Available?
PingOne Advanced Identity Cloud	Yes
PingAM (self-managed)	Yes
Ping Identity Platform (self-managed)	Yes

Inputs

All shared node state properties listed in `Mapping to server claims` are valid optional inputs to this node.

To allow the node to validate that an Identity Assertion JWT is the result of an identity request, the nonce must be present in the shared node state as `identityAssertionNonce`. This isn't required for the initiating authentication request.

Dependencies

The Identity Assertion node relies on the following:

- An Identity Assertion service must be configured globally or in the same realm, with at least one server configuration that can be selected for use with the Identity Assertion node.
- The Identity Assertion service server must have a valid shared secret encryption key configured in a secret store.

- The Identity Assertion server must be deployed, running, and accessible to the Identity Assertion node.

  It must also be configured with the shared secret encryption key.

  PingGateway can fulfil the role of the Identity Assertion server.

To use the Identity Assertion node in your AM environment, you must complete the following steps, as described in detail in the worked Example :

- Create and import a secret encryption key
- Configure the Identity Assertion service
- Map the secret label to the encryption key
- Configure PingGateway as an Identity Assertion Server

Configuration

The configurable properties for this node are:

Property	Usage
Node name	The name given to this node in the Journey. Default: `Identity Assertion` .
Identity Assertion server ID	The ID of the Identity Assertion server that handles assertion requests. The ID is composed of the server's ID and realm (if realm-scoped).
Mapping to server claims (optional)	Mapping of: • Key: Shared node state key • Value: Identity request JWT claim Required only if the server requires additional data. When a shared node state attribute has a value for a mapped key, the value is added to the identity request JWT claims according to the corresponding claim.
Mapping from server result (optional)	Mapping of: • Key: Identity Assertion JWT claim • Value: Shared node state key Required only if the server requires additional data. Default: the JWT `principal` claim is mapped to the shared node state `username` attribute. When an Identity Assertion JWT claim has a value for a mapped claim, the value is added to the shared node state according to the corresponding shared node state key.

Outputs

Any data mapped from the claims returned by the Identity Assertion server stored in the shared node state of the journey.

Successful Identity Assertion

The configuration `Mapping from server result (optional)` determines the shared node state property to set for the mandatory claim `principal`. The value of the shared node state property is set with the value of the `principal` claim.

For example, if `principal` is mapped to `usernameReceived`, the attribute `usernameReceived` is set in the shared node state. By default, `principal` is mapped to `username`.

Other values mapped in `Mapping from server result (optional)` are set in the shared node state only if the claim exists in the resulting Identity Assertion JWT.

Failed Identity Assertion

The shared node state property `error` is set with the value of the `error` claim in the resulting Identity Assertion JWT.

Outcomes

Success

The Identity Assertion server indicates that authentication was successful. It provides the authenticated `principal`.

`Error`

The Identity Assertion server indicates that authentication failed. It provides information about the error.

Troubleshooting

If the node logs an error, review the log to find the reason for the error.

Example

The following worked example describes how to use the Identity Assertion node to authenticate internal access.

Create and import a secret encryption key

Identity Assertion in AM and PingGateway uses a single secret for all encryption and decryption:

- AM uses the key to encrypt the identity request JWT; PingGateway uses it to decrypt the identity request JWT.
- PingGateway uses the key to encrypt the resulting Identity Assertion JWT; AM uses it to decrypt the Identity Assertion JWT.

Provide the encryption key in PEM format, as a JWK, or in a keystore. For example, create and import an AES PEM key into a secret store.

Configure the Identity Assertion service

Enable the service

1. In the AM admin UI, go to **Configure** > **Global Services** > **Identity Assertion Service**. Alternatively, to add the service for a realm, go to **Realms** > ***Realm name*** > **Services**, click **+Add a Service** and select **Identity Assertion Service** to create.
2. In the Identity Assertion Service page, ensure **Enable** is selected.

Configure a server

1. In the `Secondary Configurations` tab, click **+Add a Secondary Configuration** and enter the following information:
   - **Name**: A unique name for the Identity Assertion server. For example, use `IG01`.
   - **Identity Assertion server URL**: The Identity Assertion server URL. For example, enter `https://ig.ext.com:8443`.
   - **Shared Encryption Secret**: AM uses this identifier to create a secret label for encrypting the identity request JWT and resulting Identity Assertion JWT.

     The secret label takes the form `am.services.identityassertion.service.identifier.shared.secret` where identifier is the value of **Shared Encryption Secret**. For example, use identifier `idassert` to create a label called `am.services.identityassertion.service.idassert.shared.secret`.
2. Click **Create**.
3. Keep the default values for **JWT TTL (seconds)** and **Skew Allowance (seconds)** and save your changes.

Learn more about the service configuration in Identity Assertion service.

Map the secret label to the encryption key

To map the encryption key in the secret store, follow the steps in Map and rotate secrets using these values:

- **Secret Label**: Find the secret label to map by entering the value of the **Shared Encryption Secret** you used in the service configuration.

  For example, enter `idassert` to find `am.services.identityassertion.service.idassert.shared.secret`.

  You can find and configure the secret only after you have entered it in the **Shared Encryption Secret**.
- **Aliases**: Enter the alias to the encryption key secret you created earlier.

Configure PingGateway as an Identity Assertion Server

Configure PingGateway to:

- Validate the identity request JWT.
- Create an encrypted Identity Assertion JWT to send back to AM.

The PingGateway configuration includes two routes:

Authentication filter route

Directs unauthenticated requests to an authentication journey in AM.

For testing purposes, configure AM and PingGateway as described in Cross-domain single sign-on. The setup configures a demo user and validation service required for the example.

In `cdsso.json`, the CrossDomainSingleSignOnFilter uses AM's default authentication service. Add the property `authenticationService` to the CrossDomainSingleSignOnFilter to direct requests to the journey.

The following example redirects unauthenticated requests to a journey called `IgCallout`.

```
{
  "name": "CrossDomainSingleSignOnFilter-1",
  "type": "CrossDomainSingleSignOnFilter",
  "config": {
    ...
    "authenticationService" : "IgCallout",
    ...
  }
```

Identity Assertion service route

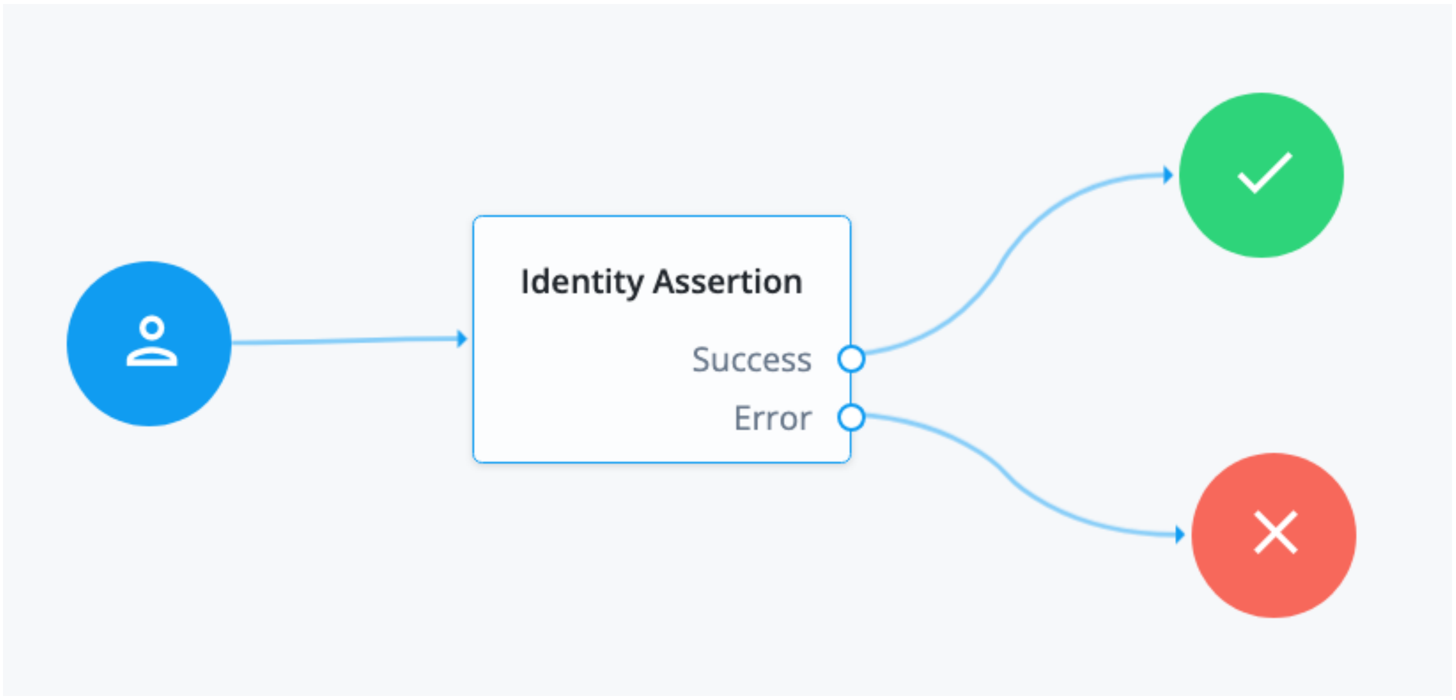
Directs unauthenticated requests to a local authentication service such as Kerberos or Windows Desktop SSO.

Consider the example in PingGateway's Example Identity Assertion service route for Identity Assertion node. The route contains an `IdentityAssertionHandler` that calls a `ScriptableIdentityAssertionPlugin` to manage local authentication.

The route requires the following:

- The key and AM setup described in this worked example.
- That the `IdentityAssertionHandler`'s `peerIdentifier` property refers to the host:port part of the deployment URL.
- That the `IdentityAssertionHandler`'s `condition` refers to the same path as the `Route` configured in the node. In this example, it refers to `/idassert`.

Configure the example authentication journey



Configure the Identity Assertion node as follows:

- **Identity Assertion server ID**: Select the ID and realm configured for the PingGateway server that supports Identity Assertion. For example, enter `IG01 [/alpha]`, where `IG01` is the name of the server created in the Configure the Identity Assertion service.
- **Route**: Enter the value of the `condition` property in the PingGateway route that will handle Identity Assertion requests. For example, enter `/idassert`, as used for the example route in Configure PingGateway as an Identity Assertion Server.

  When a request matches the path `/idassert`, the journey accesses the PingGateway route in PingGateway's Example Identity Assertion service route for IdentityAssertionNode.