Auth node reference

July 4, 2025

AM

Version: 7.4

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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

- 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)
- 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"

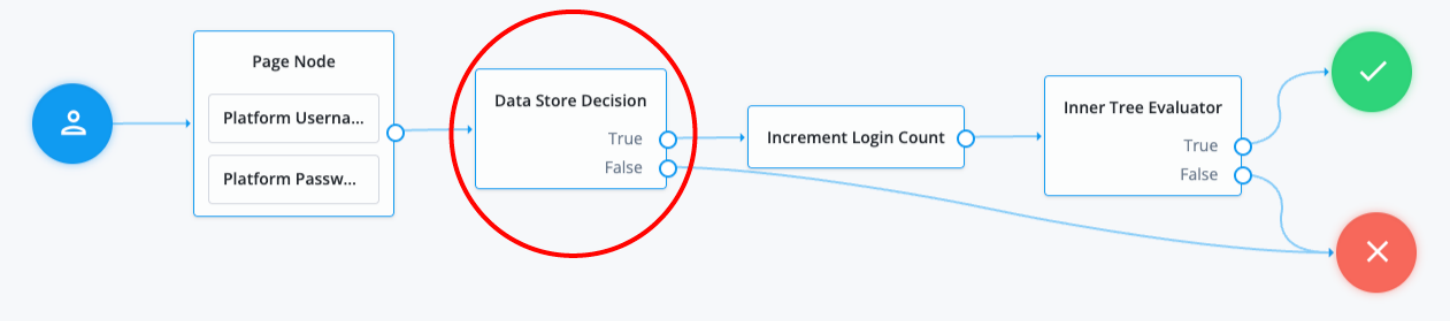
Troubleshooting

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

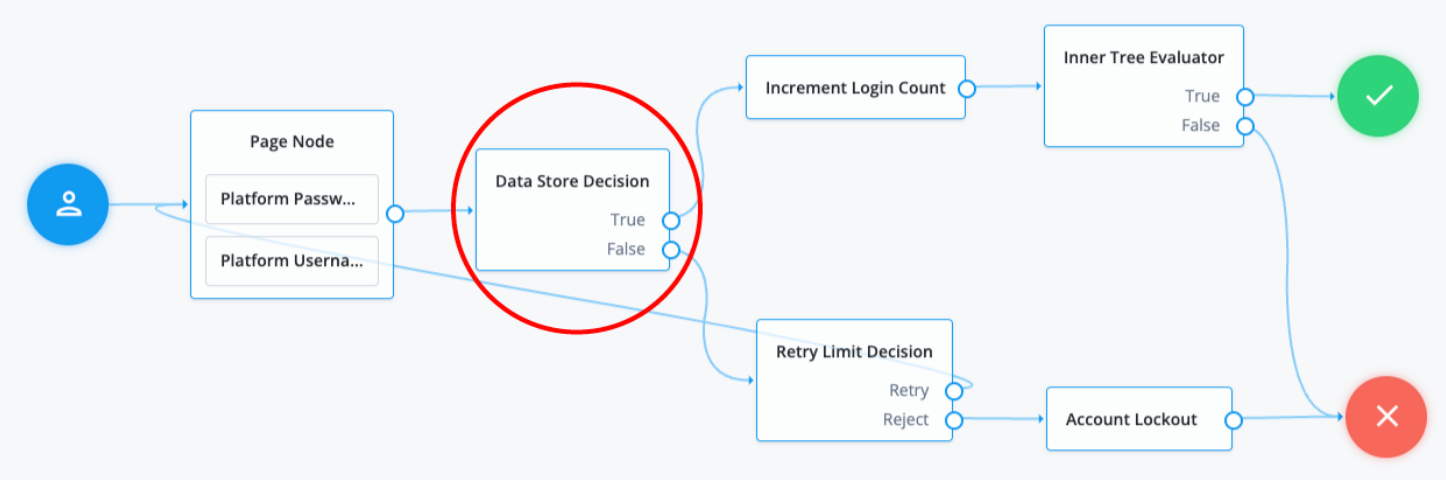
Example 1: Simple username and password collector nodes with Data Store Decision node



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.

Additional information

The following are alternate nodes that you can use in your journeys depending on your specific use cases:

- 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓

Product	Compatible?
Ping Identity Platform (self-managed)	✓

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.

Errors

The error depends on the **Authentication > Settings > Account Lockout > Login Failure Lockout Mode** setting for the realm in the AM admin UI.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `True`
- `False`

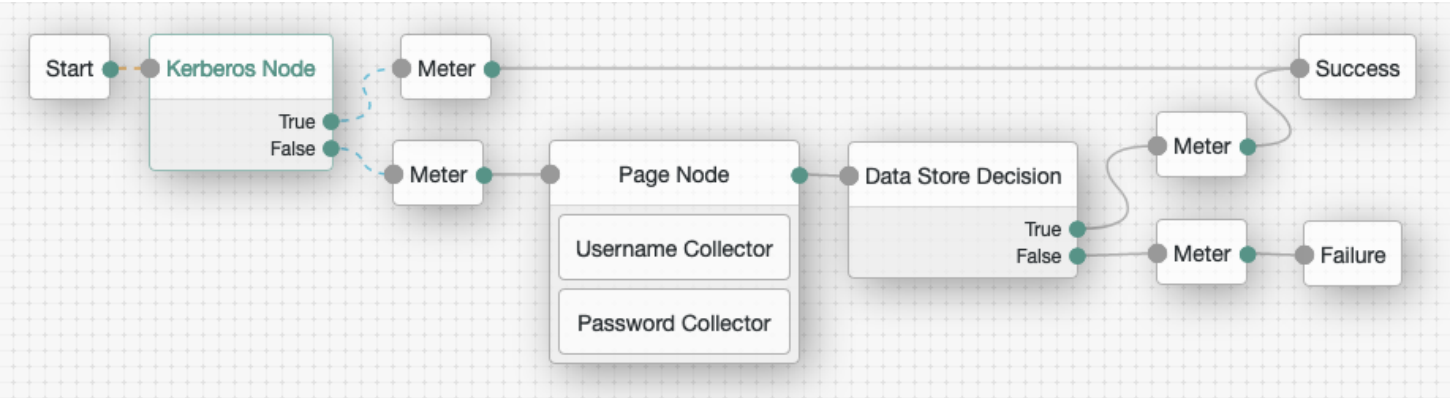
Evaluation continues along the `True` path if Windows Desktop SSO is successful; otherwise, evaluation continues along the `False` path.

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.
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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Inputs

The node reads the `username` and `password` fields from the node state.

The journey can provide these credentials in a number of ways, for example, with a combination of the Username Collector node and Password Collector node (standalone AM), Platform Username node and Platform Password node (Ping Identity Platform deployment), or by using 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`.
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 are 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.

Property	Usage
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`
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**. Default: Disabled
mTLS Secret Label	Label used to create a secret ID for mapping to the mTLS certificate in the secret store. AM uses this label to create a specific secret ID for this node. The secret ID takes the form `am.authentication.nodes.ldap.decision.mtls.label.cert` , where `label` is the value of **mTLS Secret Label**. The label 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

Property	Usage
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
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

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 ForgeRock Directory Services. 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✕

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✕

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. For an example of this layout, refer to 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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**
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

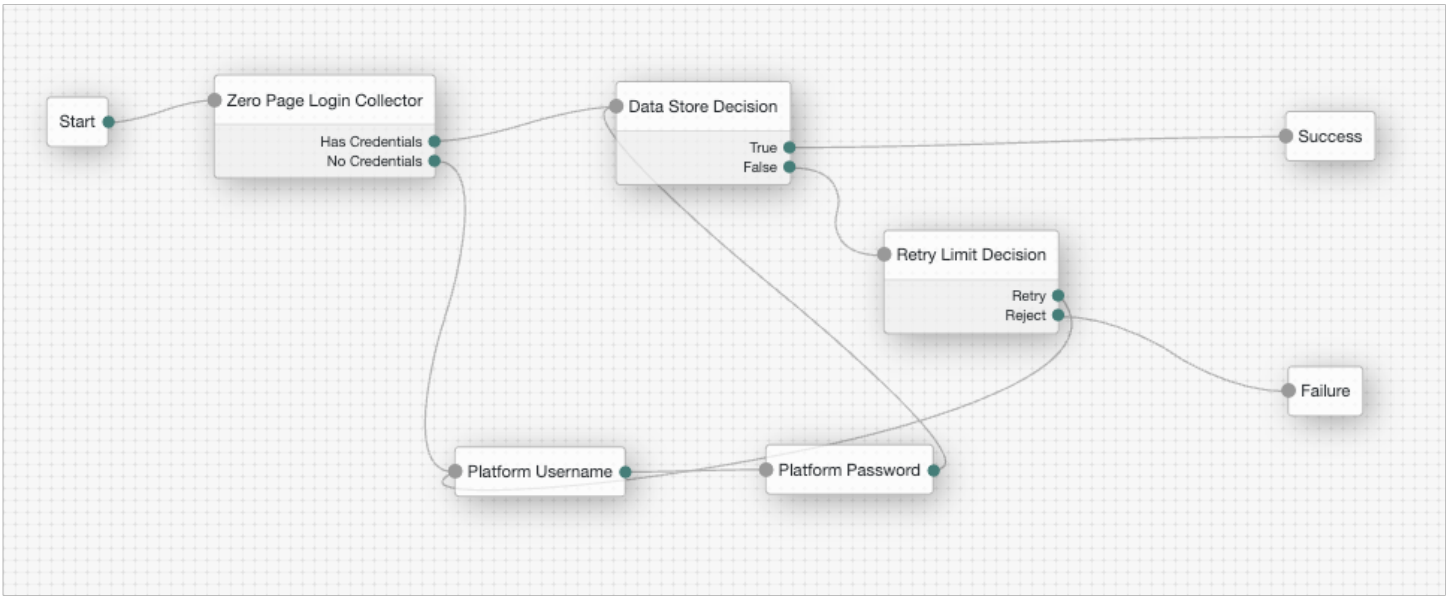
If more than one header value exists for username and/or password, the node returns the following error message

"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



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 use the Push Sender node to verify possession of the registered device. If push doesn't succeed, for example, if the user's device doesn't have internet access, the journey can fall back to the OATH Token Verifier node. to request a one-time password using OATH.

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

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Inputs

The node reads the `username` of the identity that is registering for MFA from the shared state. Implement a Username Collector node (standalone AM) or Platform Username node (Ping Identity Platform deployment)] before this node in the journey.

Dependencies

This node requires that you configure the *Push Notification Service*.

Learn more about provisioning the credentials required by the Push Notification Service in How To Configure Service Credentials (Push Auth, Docker) in Backstage in the *ForgeRock Knowledge Base*.

For detailed information about the available properties, refer to Push Notification Service.

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.
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

Property	Usage
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
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`

Property	Usage

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 } }`

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.

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.

Errors

No username found

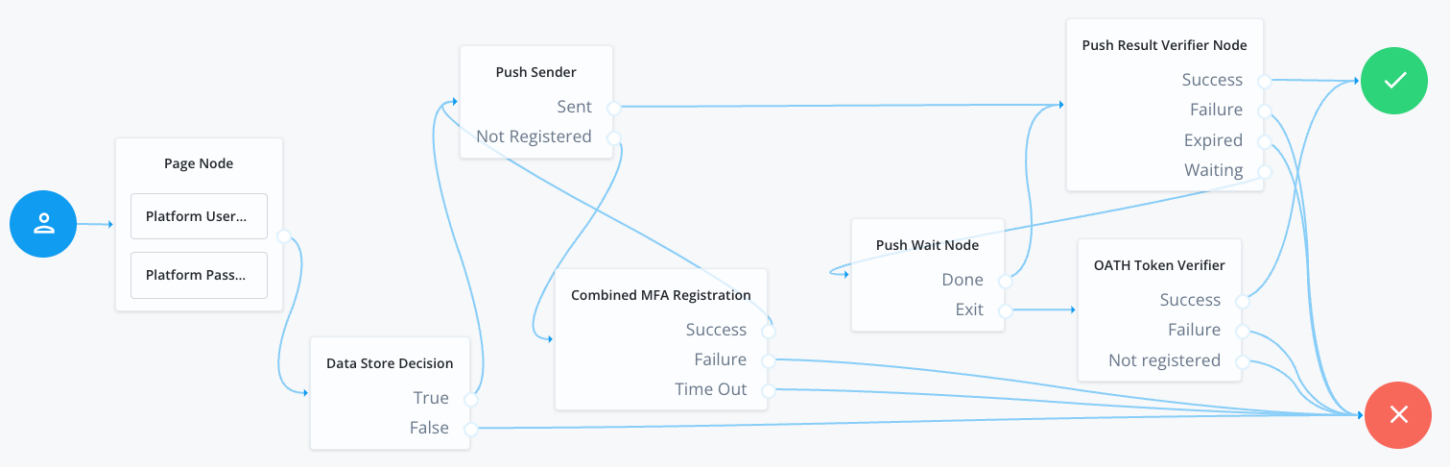
The node failed to read the username from the shared state.

Unable to find push message ID in sharedState

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

Example

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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, have not 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 Certificate extension data schema in the Google Developer 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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 Ping 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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

Displays information to obtain an authenticator application from the Apple App Store or the Google Play Store.

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>
```

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

Property	Usage
Get App Authenticator Message	Localized title for the node. The key is the language, such as `en` or `fr`, and the value is the message to display. Default: `Get the app from the {{appleLink}} or on {{googleLink}}.`
Continue Label	Localized text to use on the Continue button. The key is the language, such as `en` or `fr`, and the value is the message to display.
Apple App Store URL	Specifies the URL to download your authenticator application from the Apple App Store. The default value points to the ForgeRock Authenticator application for iOS. Default: `https://itunes.apple.com/app/forgerock-authenticator/id1038442926`
Google Play URL	Specifies the URL to download your authenticator application from the Google Play Store. The default value points to the ForgeRock Authenticator application for Android. Default: `https://play.google.com/store/apps/details?id=com.forgerock.authenticator`

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

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

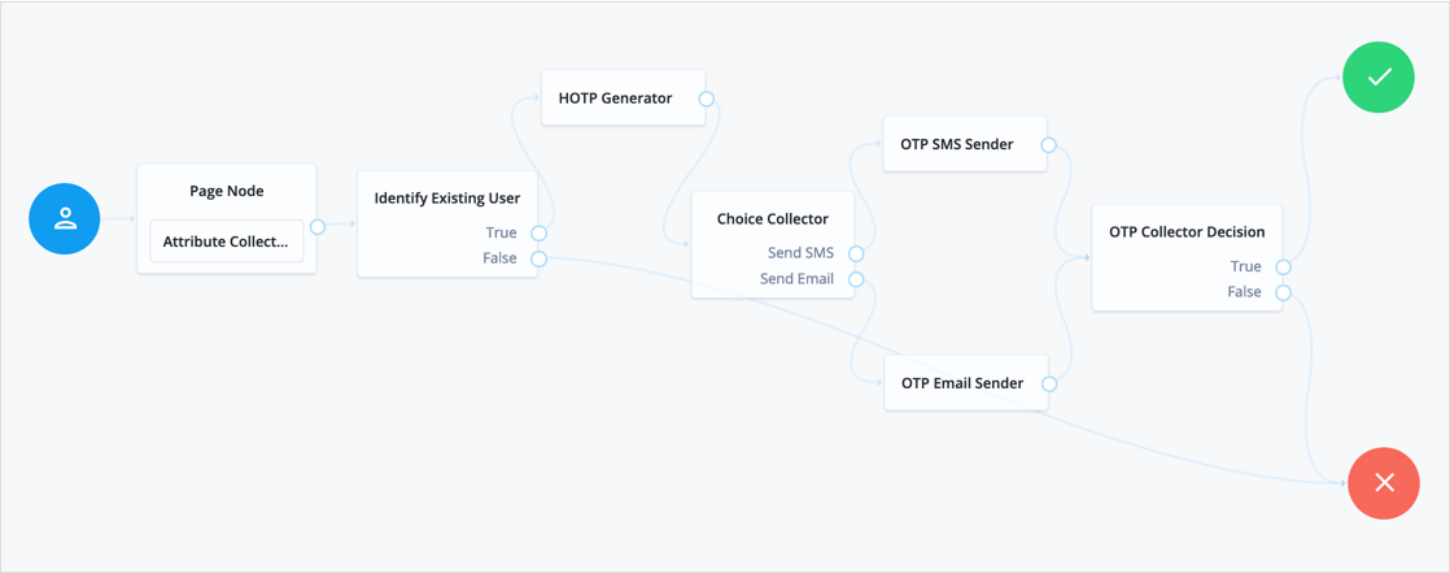
Single outcome path.

Properties

Property	Usage
One-time password length	Specify the number of digits in the one-time password. Default: 8

Example

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



MFA Registration Options node

Lets the user register a multi-factor authentication device or skip the registration process.

The node requires the username of the identity to update and the type of MFA device. For example, you can use a Username Collector node (standalone AM) or Platform Username node (Ping Identity Platform deployment) and a Push Sender node earlier in the flow to obtain these.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `Register`
- `Get App` (configurable)
- `Skip` (configurable)
- `Opt-out` (configurable)

Evaluation continues along the outcome the user selects.

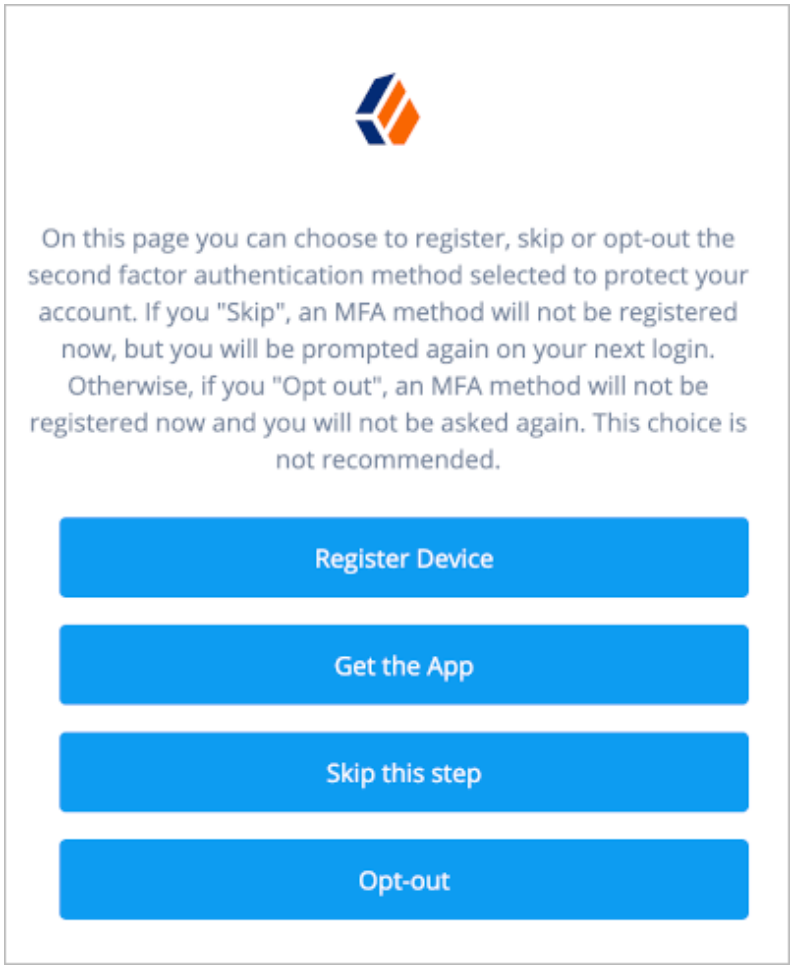
Properties

Property	Usage
Remove 'skip' option	If checked, users can no longer skip the node and must interact with it.
Display Get Authenticator App	If enabled, display the **Get the App** button.
Message	Localized text to use as the title of the screen. The key is the language, such as `en` or `fr`, and the value is the message to display.
Register Device	Localized text to use on the **Register Device** button. The key is the language, such as `en` or `fr`, and the value is the message to display.
Get Authenticator App	Localized text to use on the **Get Authenticator App** button. The key is the language, such as `en` or `fr`, and the value is the message to display.

Property	Usage
Skip this Step	Localized text to use on the **Skip this Step** button. The button and the outcome only appear if the **Remove 'skip' option** is not enabled. The key is the language, such as `en` or `fr`, and the value is the message to display.
Opt-out	Localized text to use on the **Opt-Out** button. The button and the outcome only appear if the **Remove 'skip' option** is not enabled. Note that this node does not change the user's profile. Connect the **Opt-out** outcome to an Opt-out Multi-Factor Authentication node to persist the option in the user's profile. The key is the language, such as `en` or `fr`, and the value is the message to display.

Example

Refer to the Push authentication example journey for how to use the MFA Registration Options node in a journey handling push devices.



OATH Device Storage node

Stores devices in the user profile added to the shared node state by the OATH Registration node when its **Store device data in shared state** is enabled.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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)

Outcomes

- `Success`
- `Failure`

Properties

This node has no configurable properties.

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 use with both push notifications and one-time password (OATH) verification in a single step.

The node requires the credentials of the user; for example, by using a sequence of the following nodes earlier in the authentication journey:

- 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

Connect the OATH Registration node's `Success` outcome path to the OATH Token Verifier node to continue with OTP verification.

Refer to the OATH Token Verifier node example that demonstrates how use to use other MFA nodes to create a complete OATH authentication journey.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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)

Outcomes

- `Success`
- `Failure`

If registration is successful and the device details are stored, evaluation continues along the `Success` outcome path.

If AM encounters an issue during the registration process or the user fails to complete registration, evaluation proceeds along the `Failure` path.

Properties

Property	Usage
Issuer	Specify an identifier to appear on the user's device, such as a company name, a website, or an AM realm. The authenticator application displays the value.
Account Name	Define 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.
Background Color	The background color in hex notation that displays behind the issuer's logo within the authenticator application.
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.
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.
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.
Minimum Secret Key Length	Number of hexadecimal characters allowed for the Secret Key.
OATH Algorithm	Specify the algorithm your device uses to generate the OTP: ***HOTP*** HOTP uses a counter value that is incremented every time a new OTP is generated. ***TOTP (default)*** TOTP generates a new OTP every few seconds as specified by the `TOTP Time Step Interval` value. If this is set to HOTP, set the same value in the OATH Token Verifier node.

Property	Usage
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. The default value is 30.
TOTP Hash Algorithm	The HMAC hash algorithm used to generate the OTP codes. AM supports SHA1, SHA256, and SHA512.
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.
HOTP Truncation Offset	This is an option used by the HOTP algorithm that not all devices support. Leave the default value of -1 unless you know user devices use an offset.
QR code message	The message with instructions to scan the QR code to register the device. Click **Add**. Enter the message locale in the `Key` field; for example, `en-gb`. Enter the message to display to the user in the `Value` field.

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.

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 that the user credentials are authenticated, and that the user has previously registered a device using the OATH Registration node. These two nodes work together to provide all the capabilities of a secure OATH authentication journey.

You can also use them with other MFA nodes such as the following to extend these capabilities:

- Get Authenticator App node
- MFA Registration Options node
- Opt-out Multi-Factor Authentication node

> **Note**
>
> You can use the OATH nodes in conjunction with the ForgeRock Authenticator application to register your phone, receive notifications, or generate one-time passwords.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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)

Outcomes

Evaluation continues along one of the following outcome paths:

`Success`

There is a registered device and the token code is verified.

`Failure`

The user is not authenticated, or the collected token code cannot be verified.

`Not registered`

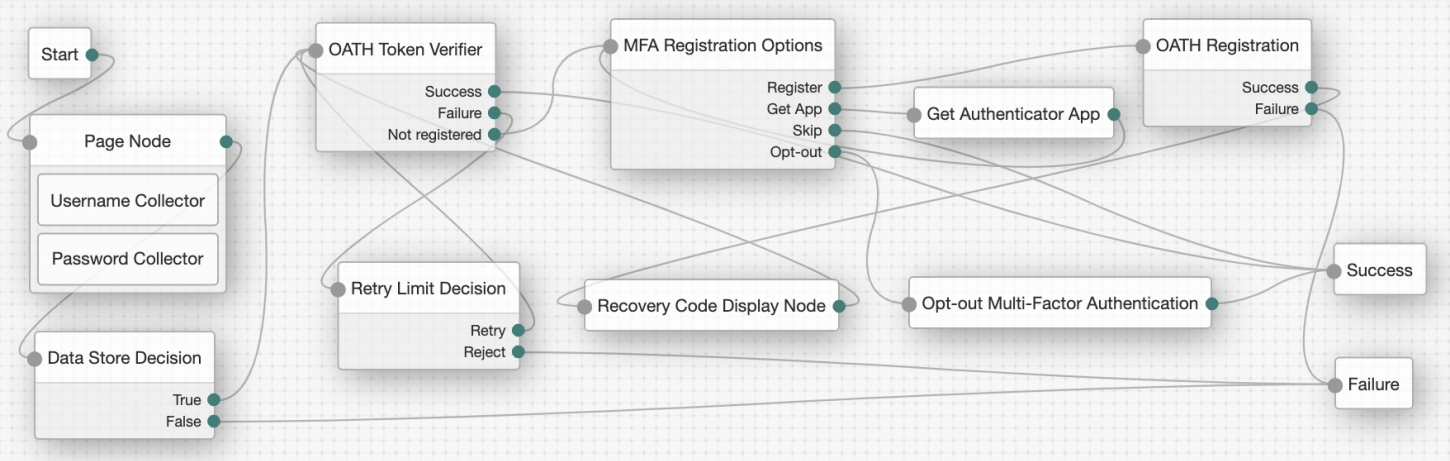
There is no registered device for the user.

Properties

Property	Usage
OATH Algorithm	Specify the algorithm your device uses to generate the OTP: ***HOTP*** HOTP uses a counter value that is incremented every time a new OTP is generated. ***TOTP (default)*** TOTP generates a new OTP every few seconds as specified by the `TOTP Time Step Interval` value. If this is set to HOTP, you need to set the same value in the OATH Registration node.
HOTP Window Size	This property sets the window that 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. The default value is 100.
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. The default value is 30.
TOTP Time Steps	This is the number of time step intervals that the OTP is permitted to be out of sync. This applies to codes that are generated before or after the current code. For example, with a time step of 1, the server permits either the previous, the current, or the next code. The default value is 2.
TOTP Hash Algorithm	The HMAC hash algorithm to be used to generate the OTP codes. ForgeRock Authenticator (OATH) supports SHA1, SHA256, and SHA512.
TOTP Maximum Allowed Clock Drift	Number of time steps a client can be out of sync with the server before manual resynchronization is required. For example, with 3 allowed drifts and a time step interval of 30 seconds, the server allows codes from up to 90 seconds from the current time to be treated as 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 user's authentication code is rejected. The default value is 5.

Property	Usage
Allow recovery codes	Specify whether to allow users to use one of the recovery codes to proceed with the login.

Example



Opt-out Multi-Factor Authentication node

Sets the `skippable` attribute in the user's profile, which lets them skip MFA.

The node requires the username of the identity to update, and the type of MFA device. For example, you can use a Username Collector node (standalone AM) or Platform Username node (Ping Identity Platform deployment) and a Push Sender node earlier in the flow to obtain these.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Evaluation continues along the single outcome path after setting the MFA device as `skippable` in the user's profile.

Properties

This node has no configurable properties.

OTP Collector Decision node

Requests and verifies one-time passwords.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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

Sends an email containing a generated one-time password 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 particular instance, go to **Deployment > Servers > *Server Name* > Advanced**.
- To configure advanced server properties for a particular instance, go to **Deployment > Servers > *Server Name* > Advanced**.

If the property you want to add or edit is already configured, click on the pencil (✎) button to edit it. When you are finished, click on the tick (✓) button.
Save your changes.
For more information, refer to Advanced Properties.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

Property	Usage
Mail Server Host Name *(required)*	Specifies the hostname of the SMTP email server.
Mail Server Host Port	Specifies the outgoing mail server port. Common ports are 25, 465 for SSL/TLS, or 587 for StartTLS.
Mail Server Authentication Username	Specifies the username AM uses to connect to the mail server.
Mail Server Authentication Password	Specifies the password AM uses to connect to the mail server.

Property	Usage
Email From Address *(required)*	Specifies the email address from which the one-time password will appear to have been sent.
Email Attribute Name	Specifies the user's profile attribute containing the email address to which to email the OTP. 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	Specifies how to connect to the mail server. If a secure method is specified, 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	Specifies 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`

OTP SMS Sender node

Uses an email-to-SMS gateway provider to send an SMS message containing a generated one-time password 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`.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

Property	Usage
Mail Server Host Name *(required)*	Specifies the hostname of the SMTP email server.
Mail Server Host Port	Specifies the outgoing mail server port. Common ports are 25, 465 for SSL/TLS, or 587 for StartTLS.
Mail Server Authentication Username	Specifies the username AM uses to connect to the mail server.
Mail Server Authentication Password	Specifies the password AM uses to connect to the mail server.
Email From Address *(required)*	Specifies the email address from which the one-time password will appear to have been sent.
Mobile Phone Number Attribute Name	Specifies the user's profile attribute containing the mobile phone number to which to send the SMS containing the OTP. Default: `telephoneNumber`
Mobile Carrier Attribute Name	Specifies the user's profile attribute containing the mobile carrier domain used as the email to SMS gateway.
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.

Property	Usage
Mail Server Secure Connection	Specifies how to connect to the mail server. If a secure method is specified, 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	Specifies 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`

Push Registration node

Provides a way to register a device, such as a mobile phone for multi-factor authentication using push notifications.

Find more information in MFA: Push authentication.

The node requires the username of the identity to update; for example, by using a Username Collector node (standalone AM) or Platform Username node (Ping Identity Platform deployment).

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

You must also configure the *Push Notification Service*.

For information on provisioning the credentials required by the Push Notification Service, refer to How To Configure Service Credentials (Push Auth, Docker) in Backstage in the *ForgeRock Knowledge Base*.

For detailed information about the available properties, refer to Push Notification Service.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓

Product	Compatible?
Ping Identity Platform (self-managed)	✓

Authenticators

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

Third-party authenticator apps are not compatible with ForgeRock's push notification functionality.

Outcomes

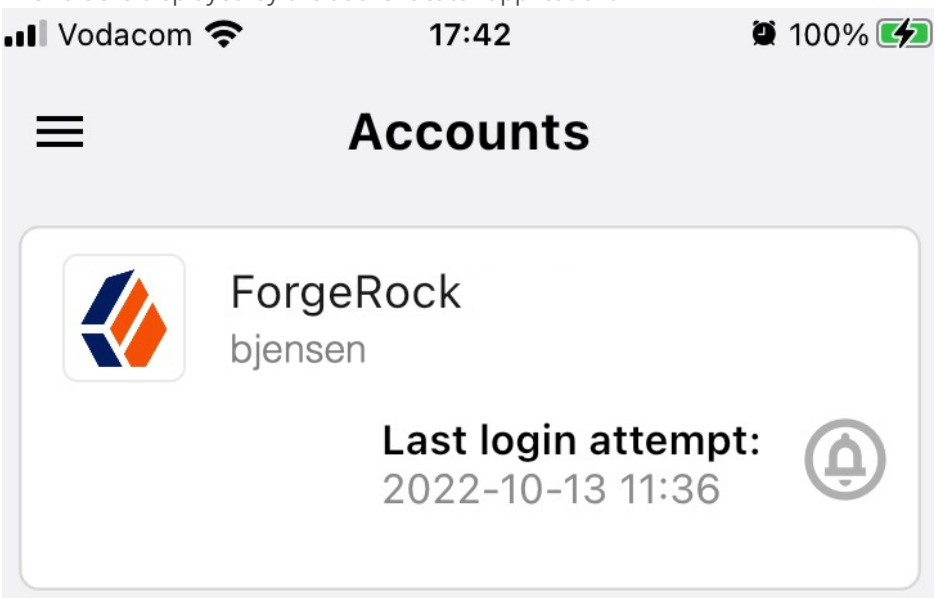
- `Success`
- `Failure`
- `Time Out`

If the user successfully registers their authenticator, evaluation continues along the `Success` outcome path.

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

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

Properties

Property	Usage
Issuer	Specify an identifier so that the user knows which service their account relates to. The value is displayed by the authenticator application:  For example, `Example Inc.` or the name of your application.

Property	Usage
Account Name	Specifies 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.
Registration Response Timeout	Specify the number of seconds to wait for a response from the authenticator. If the specified time is reached, evaluation continues along the `Time Out` outcome path.
Background Color	Specifies the background color, in hex notation, to display behind the issuer's logo within the ForgeRock Authenticator application.
Logo Image URL	Specifies the location of an image to download and display as the issuer's logo in the ForgeRock Authenticator application.
Generate Recovery Codes	Specify whether push-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. **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.
QR code message	The message with instructions to scan the QR code to register the device. Click **Add**. Enter the message locale in the **Key** field; for example, `en-gb`. Enter the message to display to the user in the **Value** field.

Example

Refer to the Push authentication example journey for how to use the Push Registration node in a journey handling push devices.

Push Result Verifier node

Works with the Push Sender node to validate the user's response to a previously sent push notification message.

Tip

If the push message contained any additional information, for example, if it was a registration request, the values are stored in the `nodeState` object on the `pushContent` key.
For information on creating or customizing authentication nodes, refer to Node development.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Authenticators

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

Third-party authenticator apps are not compatible with ForgeRock's push notification functionality.

Outcomes

- `Success`
- `Failure`
- `Expired`
- `Waiting`

Evaluation continues along the `Success` outcome path if the push notification was approved by the user.

Evaluation continues along the `Failure` outcome path if the push notification was rejected by the user.

If no response to the push notification was received within the Message Timeout value specified in the Push Sender node, evaluation continues along the `Expired` outcome path.

If no response to the push notification has been received yet, evaluation continues along the `Waiting` outcome path.

Properties

This node has no configurable properties.

Push Sender node

Sends push notification messages to a device for multi-factor authentication.

Configure the AM Push Notification Service for the realm before using this node. For information on the properties used by the service, refer to Push Notification Service.

For information on provisioning the credentials used by the service, refer to How To Configure Service Credentials (Push Auth, Docker) in Backstage in the *ForgeRock Knowledge Base*.

To determine whether the user has a registered device, the flow must have included the username in the shared state, for example, by using a Username Collector node (standalone AM) or Platform Username node (Ping Identity Platform deployment).

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Authenticators

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

Third-party authenticator apps are not compatible with ForgeRock's push notification functionality.

Outcomes

- `Sent`
- `Not Registered`
- `Skipped`
- `Failure`

Evaluation continues along the `Sent` outcome path if the push notification was successfully sent to the handling service.

If the user doesn't have a registered device, evaluation continues along the `Not Registered` outcome path.

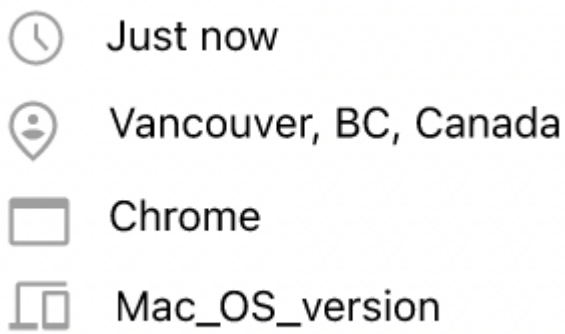
If the user chooses to skip push authentication, evaluation continues along the `Skipped` outcome path.

The node displays the `Failure` outcome only if you enable the **Capture failure** configuration option. In this case, evaluation proceeds along the `Failure` path if there is an error during execution of the node.

Properties

Property	Usage
Message Timeout	Specifies the number of milliseconds the push notification message will remain valid. The Push Result Verifier node rejects responses to push messages that have timed out.

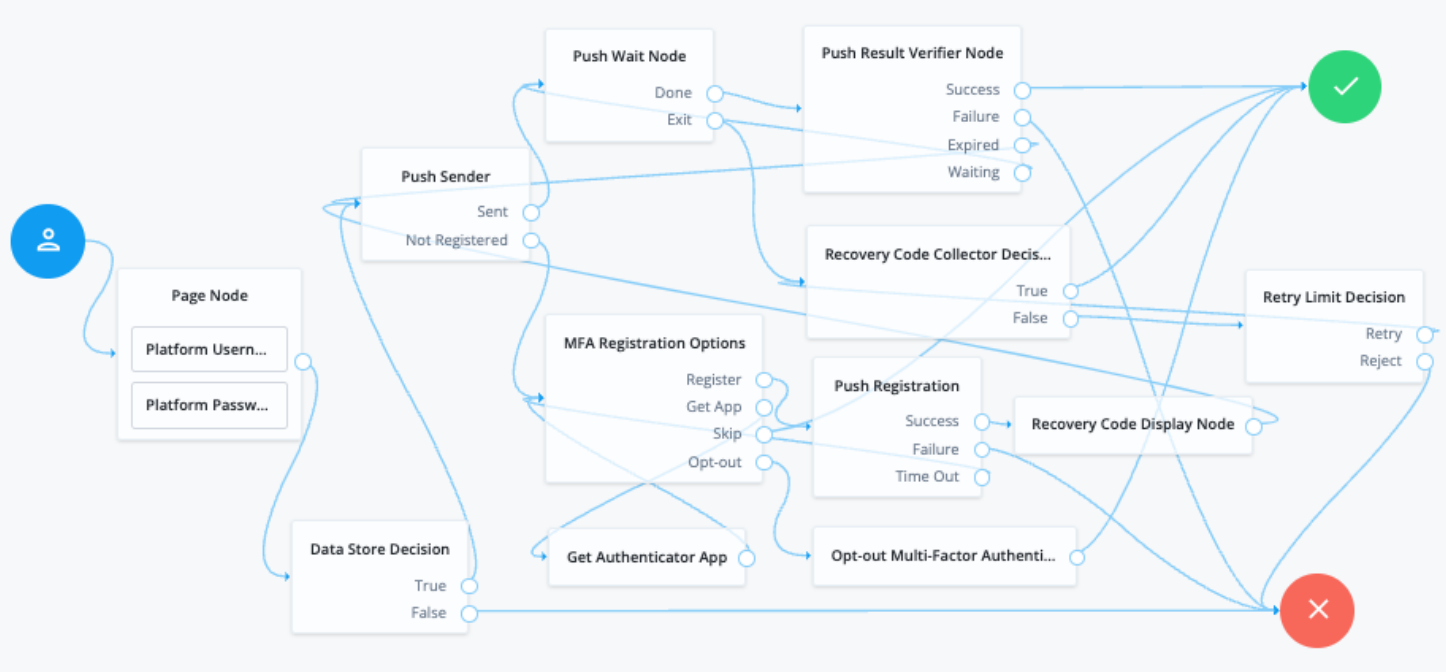
Property	Usage
User Message	Specifies the optional message to send to the user. You can provide the message in multiple languages by specifying the locale in the `KEY` field; for example, `en-US`. The locale selected for display is based on the user's locale settings in their browser. Messages provided in the node override the defaults provided by AM. For information about customizing and translating the default messages, refer to Internationalization. The following variables can be used in the `VALUE` field: **`{{user}}`** Replaced with the username value of the account registered in the ForgeRock Authenticator application, for example *Demo*. **`{{issuer}}`** Replaced with the issuer value of the account registered in the ForgeRock Authenticator application, for example, *ForgeRock*. Example: `Login attempt from {{user}} at {{issuer}}.`
Remove 'skip' option	Enable this option in the node to make the push authentication mandatory. When disabled, the user can skip the push authentication requested by the node, and evaluation continues along the `Skipped` outcome path. Default: Disabled **Note** Nodes in authentication trees aren't affected by the `Two Factor Authentication Mandatory` property (at **Realms > *Realm Name* > Authentication > Settings > General**). This property applies only to modules within authentication chains.

Property	Usage
Share Context info	If enabled, context data such as `remoteIp`, `userAgent`, and `location` are included in the notification payload. For example: `{ "location": { "latitude": 49.2208569, "longitude": -123.1174431 }, "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36", "remoteIp": "9.9.9.9" }` The ForgeRock Authenticator displays this additional information to the user to help verify that the request is genuine and initiated by them.  ***Figure 1. Context information in the ForgeRock Authenticator*** For the location attribute to be set, the flow must contain a Device Profile Collector node with **Collect Device Location** enabled.
Custom Payload Attributes	Specify shared state objects to be included in the message payload sent to the client. The size of the payload must not exceed 3 Kb or a `NodeProcessException` is thrown. To add a custom attribute, enter the shared state object name in the text field and click **Add**. Repeat for each object you want to include in the payload.

Property	Usage
Push Type	Select the type of the push authentication the user must perform on their device to continue the journey. Possible values are: **`Tap to Accept`** ***(default)*** Requires the user to tap to accept. **`Display Challenge Code`** Requires the user to select one of three numbers displayed on their device. This selected number must match the code displayed in the browser for the request to be verified. **`Use Biometrics to Accept`** Requires the user's biometric authentication to process the notification. The actions the user performs vary depending on the selected option. Refer to Respond to push notifications.
Capture failure (optional)	If enabled, and the node fails to send the Push Notification, the journey skips the node. The journey stores the reason for the failure in the `PushAuthFailureReason` key in the shared state for use by subsequent nodes in the journey. Possible failure reasons include `MISSING_USERNAME`, `SENDER_ALREADY_USED`, `CTS_ERROR`, and `TRANSMISSION_FAILURE`.

Example

The following example shows one possible implementation of multi-factor push authentication:



List of node connections

Source node	Outcome path	Target node
Page Node containing: Username Collector and Password Collector (standalone AM) or Platform Username and Platform Password (Ping Identity Platform deployment).	→	Data Store Decision
Data Store Decision	True	Push Sender
	False	Failure
Push Sender	Sent	Push Wait
	Not Registered	MFA Registration Options
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

Source node	Outcome path	Target node
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 Push Sender node.

If the *user has a registered device*:

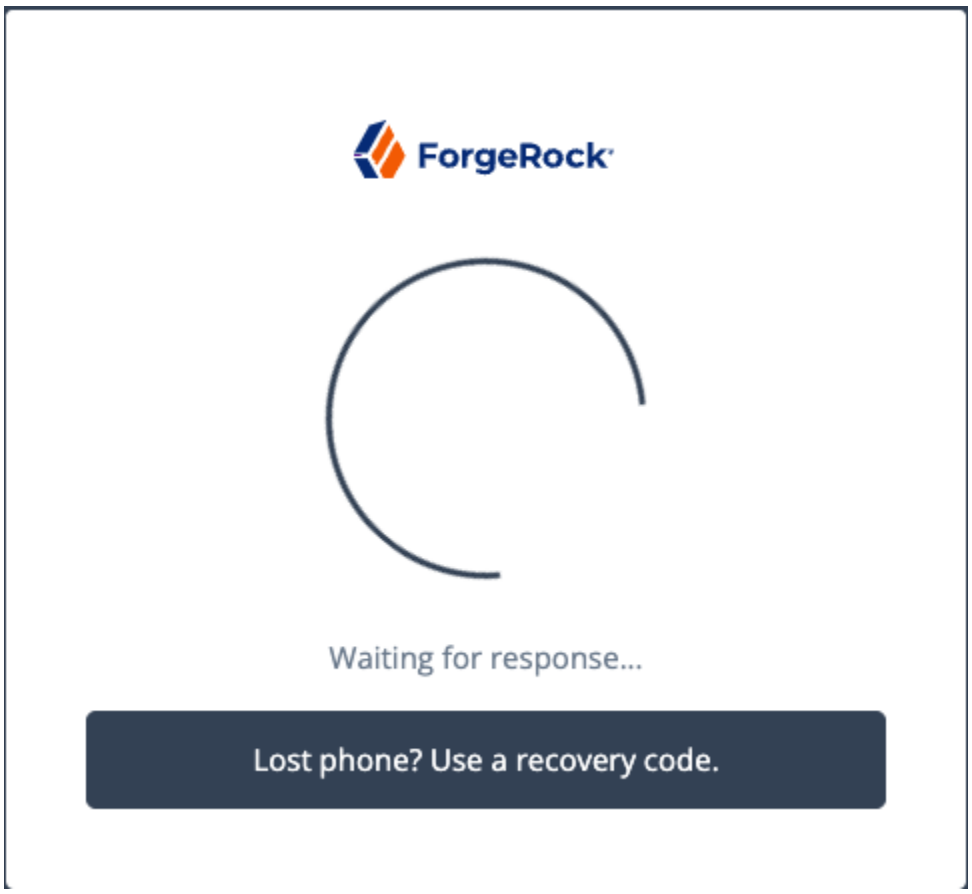
1. AM sends a push to their registered device.
2. The Push Wait node pauses authentication for 5 seconds, during which time the user can respond to the push notification on their device; for example, by using the ForgeRock Authenticator application.
    - If the user responds positively, they are authenticated successfully and logged in.
    - If the user responds negatively, they are not authenticated successfully and do not receive a session.
    - If the push notification expires, AM 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 has not yet responded, the flow loops back a step and the Push Wait node pauses authentication for another 5 seconds.

   If the user exits the Push Wait node, they can enter a recovery code in order to authenticate.

   For this situation, configure the **Exit Message** property in the Push Wait node with a message, such as `Lost phone? Use a recovery code`.



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

If the user *does not 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 the QR code that should be scanned with a suitable authenticator application.

Get the App

The flow continues to the Get Authenticator App node, which displays the links needed to obtain a suitable application, such as the ForgeRock Authenticator.

Skip this step

Displayed only if the node configuration lets the user skip. In this example, skipping is linked to the `Success` outcome. Alternatively, an Inner Tree Evaluator node could have been used for authentication.

Opt-out

Displayed only if the node configuration allows the user to skip or opt out. 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 node.

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

   After registration, the recovery codes are displayed to the user for safekeeping, and evaluation continues with the Push Sender node to start push notification.

> **Note**
>
> To manage push devices, the user must log in using either the device or a recovery code.
> For more information, refer to Manage devices for MFA.

Respond to push notifications

The default **Push Type** setting is `Tap to Accept`. This requires the user to tap to either `Accept` or `Reject` in the ForgeRock Authenticator.

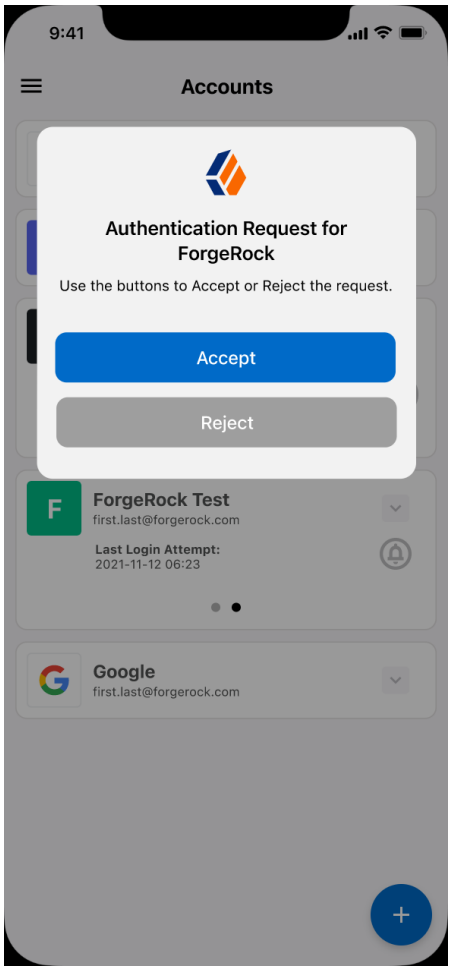


Figure 2. Tap to Accept (Default)

Research shows that users might accept a push authentication without fully checking if it is legitimate. To reduce the chances of a user accepting a malicious push authentication attempt, you can configure two additional push types:

`Display Challenge Code`

Requires the user to select one of three numbers displayed on their device. This selected number must match the code displayed in the browser for the request to be verified.

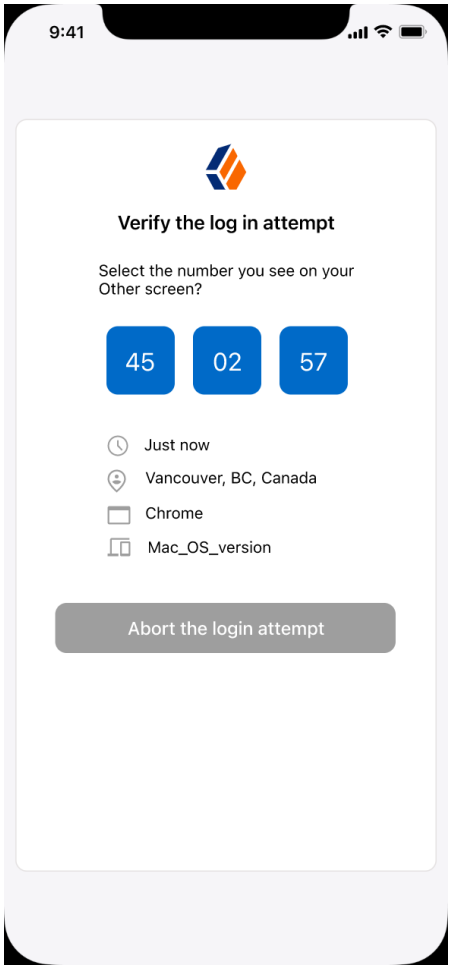


Figure 3. Challenge code

`Use Biometrics to Accept`

Requires the user's biometric authentication to process the notification, after tapping `Accept` or `Reject`.

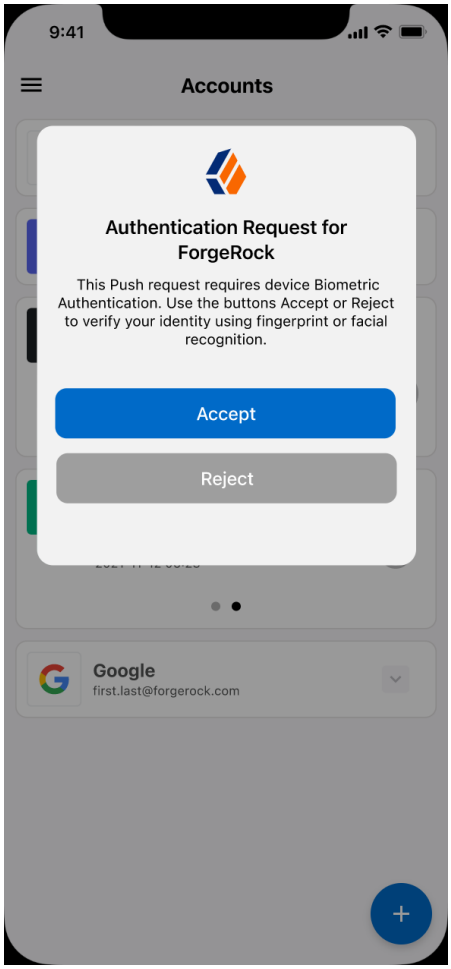


Figure 4. Biometric authentication required

Push Wait node

Pauses the authentication for the specified number of seconds during the processing of a push authentication request.

When push authentication involves a number selection challenge, where the push type of the Push Sender node is set to `Display Challenge Code`, the node displays the code challenge for the user to complete. Connect this node to a Push Result Verifier node to check the result of the code challenge.

Both nodes' waiting times and the messages are configurable.

The message displayed on the exit button can be configured using the `Exit Message` property.

To provide localized versions of the waiting, push challenge, and exit messages in multiple languages, configure the message properties to specify the locale in the `KEY` field (for example, `en-US`) and the message in the `VALUE` field. 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.

For information about customizing and translating the default messages, refer to Internationalization.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Authenticators

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

Third-party authenticator apps are not compatible with ForgeRock's push notification functionality.

Outcomes

- `Done`
- `Exit`

Evaluation continues along the `Done` outcome path after the wait time has passed. Evaluation continues along the `Exit` outcome path if the user clicks the exit button.

Properties

Property	Usage
Seconds To Wait	Specify the number of seconds to pause authentication. Default: `5`
Waiting Message	Customize the message to display to the user. To include the remaining seconds in the message, use the `{{time}}` variable. Click **Add** to enter a KEY and VALUE for a localized message and **+** to save. Repeat for each supported language. Default: `Waiting for response…`
Push Challenge Message	Customize the message containing the challenge code. To include the number challenge, use the `{{challenge}}` variable. Click **Add** to enter a KEY and VALUE for a localized message and **+** to save. Repeat for each supported language. Default: `Tap the number [{{challenge}}] on the Push Notification to continue.`

Property	Usage
Exit Message	Customize the message to display to the user when they choose to exit the node before the wait period has elapsed. The message is displayed as a link. Click **Add** to enter a KEY and VALUE for a localized message and **+** to save. Repeat for each supported language. Default: `Cancel`

Example

Refer to the Push authentication example journey for how to use the Push Wait node in a journey handling push devices.

Recovery Code Collector Decision node

Lets users authenticate with a recovery code provided when registering a device for multi-factor authentication.

Use this node for a flow 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. For more information on viewing the recovery codes when registering a device, refer to Register the ForgeRock Authenticator for multi-factor authentication.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `True`
- `False`

Evaluation continues along the `True` outcome path if the provided recovery code matches one belonging to the user. To determine whether the provided code belongs to the user, the shared state must include the username. You can obtain this using a Username Collector node (standalone AM) or Platform Username node (Ping Identity Platform deployment).

If the recovery code does not match, or a username has not been acquired, evaluation continues along the `False` outcome path.

Properties

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

Recovery Code Display node

Retrieves generated recovery codes from the transient state and presents them to the user, for safe-keeping. The codes can be used to authenticate if a registered device is lost or stolen.

Use this node with the WebAuthn Registration node, the OATH Registration node or the Push Registration node.

Generated recovery codes are inserted into transient state when evaluation continues along the `Success` outcome path of the MFA nodes configured to generate recovery codes. Connect this node to the `Success` outcome path to display the codes.

If no recovery codes are available in transient state, evaluation continues along the only outcome path, and nothing is displayed to the user.

> **Important**
>
> Generated recovery codes cannot be retrieved from the user's profile—they are one-way encrypted. This node is the one and only opportunity to view and save the recovery codes.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

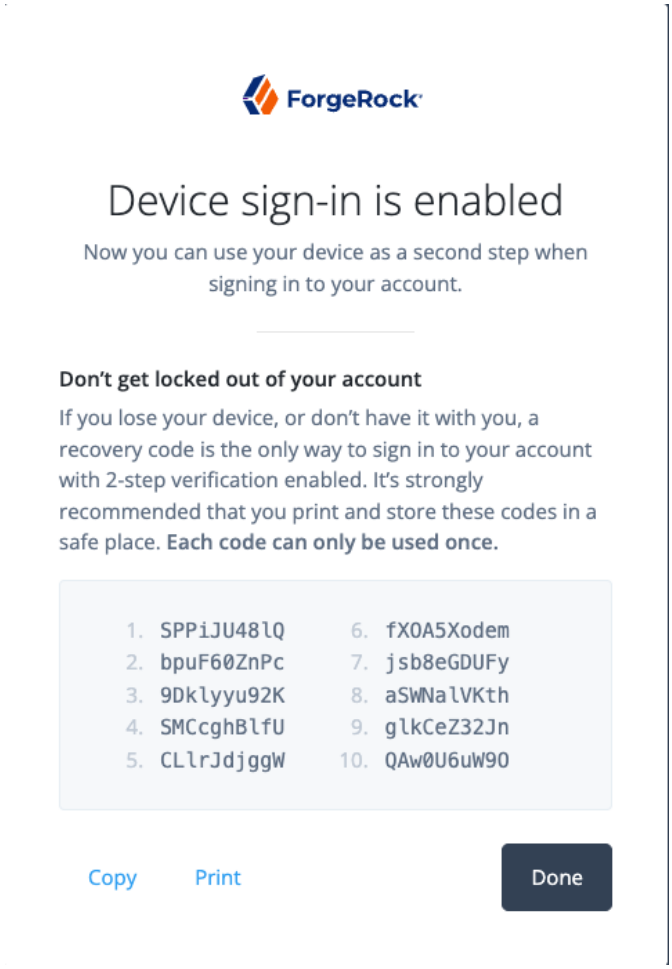
Single outcome path.

Properties

This node has no configurable properties.

Example

The following shows example output of this node:



WebAuthn Authentication node

Lets users on supported clients use a registered FIDO device during authentication.

To determine whether the user has a registered device, the node reads the `username` from the shared state. Implement a Username Collector node (standalone AM) or Platform Username node (Ping Identity Platform deployment) earlier in the journey.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `Unsupported`
- `No Device Registered`
- `Success`
- `Failure`
- `Client Error`
- `Recovery Code` (configurable)

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

If the user does not have a registered device, evaluation continues along the `No Device Registered` outcome path.

If AM encounters an issue when attempting to authenticate using the device, evaluation continues along the `Failure` outcome path. For example, AM could not verify that 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 authenticate using the device, for example, if the timeout was reached, 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 **Allow recovery code** is enabled, AM provides the user the option to enter a recovery code rather than authenticate using a device. Evaluation continues along the `Recovery Code` outcome path if the users chooses to enter a recovery code. To accept and verify the recovery code, ensure the outcome path leads to a Recovery Code Collector Decision node.

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.

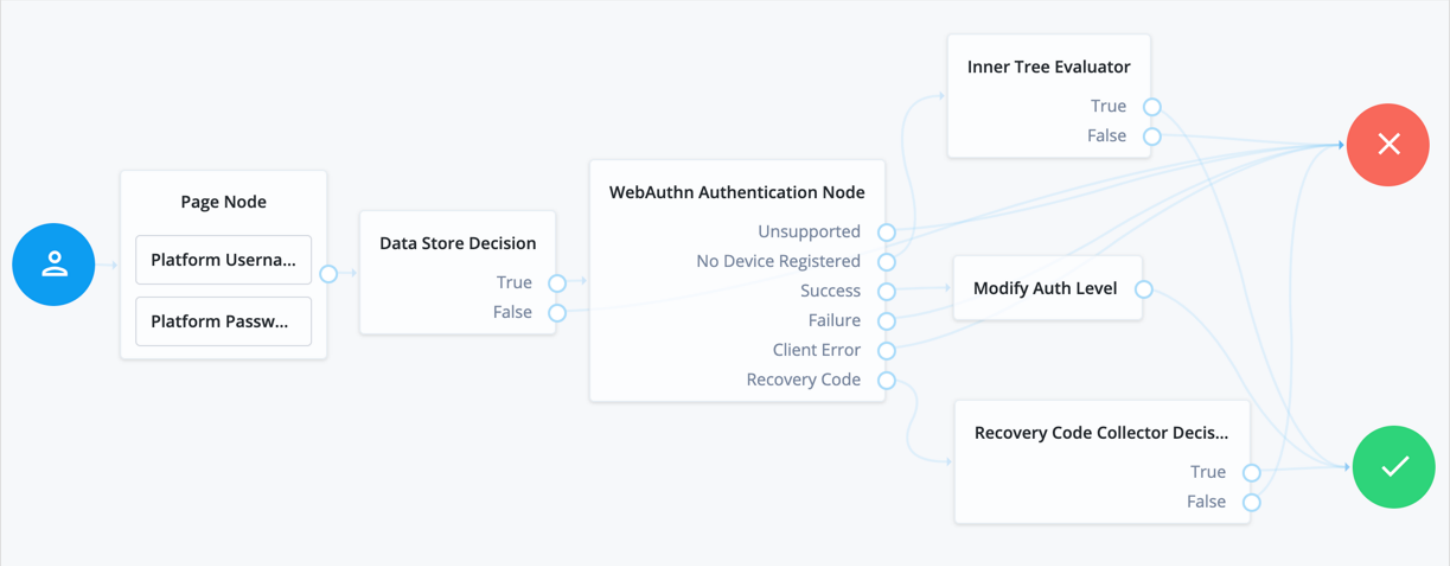
Properties

Property	Usage
Relying party identifier	Specifies the domain used as the relying party identifier during web authentication. If not specified, AM uses the domain name of the instance, for example, `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.

Property	Usage
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 authentication. **`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.
Allow recovery codes	Specify whether to allow the user to enter one of their recovery codes instead of performing an authentication gesture. Enabling this options adds a `Recovery Code` outcome path to the node. This outcome path should lead to a Recovery Code Collector Decision node to collect and verify the recovery code.
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.
Username from device	Specifies whether AM requests that the device provides the username. When enabled, if the device is unable to store or provide usernames, the node fails and evaluation continues along the `Failure` path. For information on using this property for usernameless authentication with ForgeRock Go, refer to 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.

Example

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

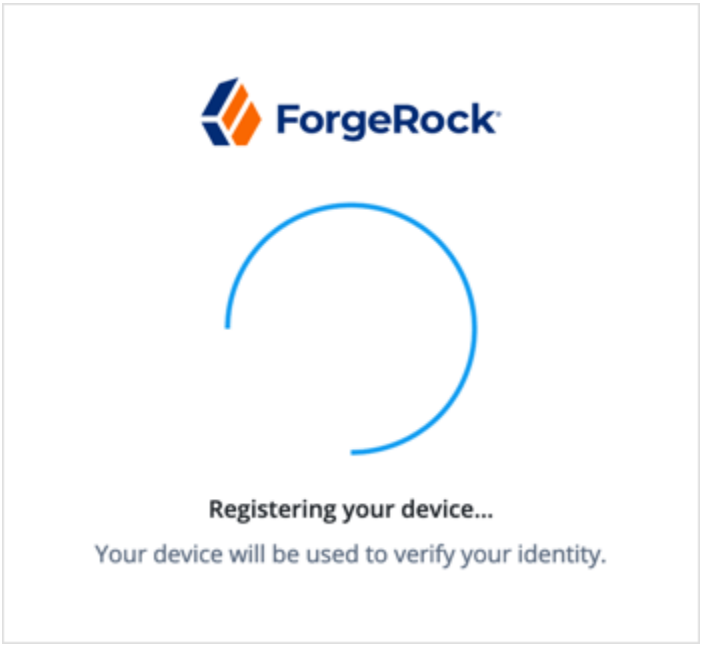


After verifying the users credentials against the configured data store, evaluation continues to the WebAuthn Authentication node.

If the user's client does not support WebAuthn, authentication fails and the user does not 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 does support 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:



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 depends 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 which verify the identity of the user. For extra protection, AM **WILL** verify the response from an authenticator matches the criteria configured for the node, and will reject—with the `Failure` outcome—an authentication attempt by an inappropriate authenticator type.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Property	Usage
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.

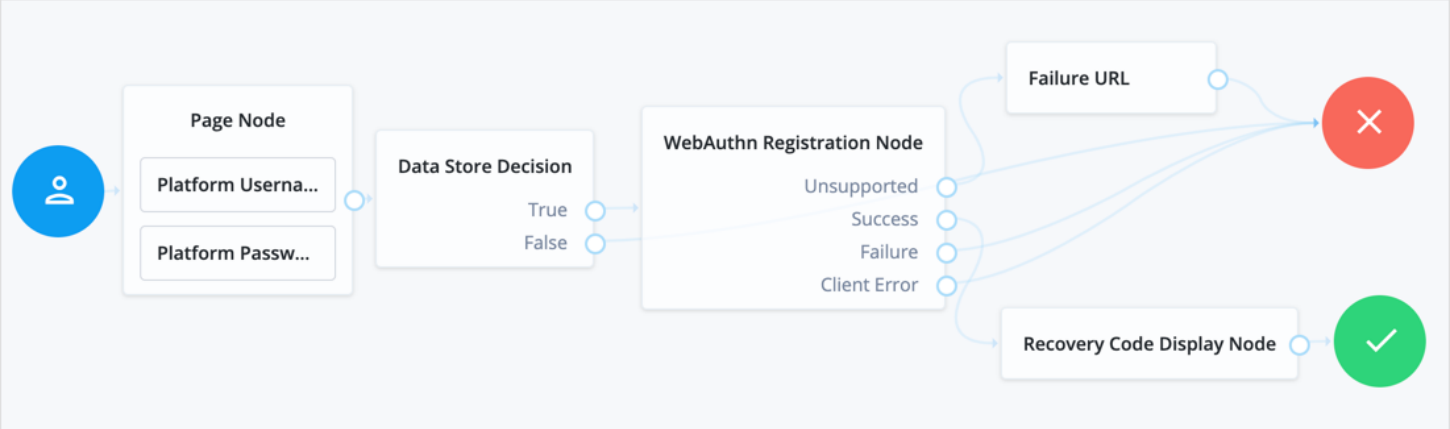
Property	Usage
Authentication attachment	Specifies whether AM requires that the authenticator is a particular attachment type. There are two types of authenticator attachments: • An authenticator that is built-in to the client device is labeled a *platform attachment*. A fingerprint scanner built-in 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. If you have enabled the **Store device data in transient state** property and are not saving the device data to the user's profile immediately, do not enable the **Generate recovery codes** property in this node, but 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. For information on using this property for usernameless authentication with ForgeRock Go, refer to 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. For information on using this property for usernameless authentication with ForgeRock Go, refer to 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.

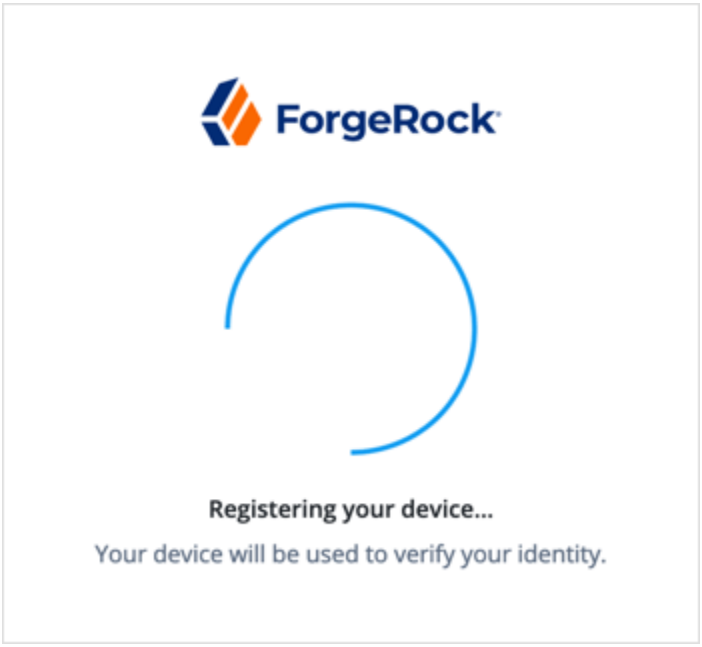
Example

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:



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

Checks whether the current account is active.

This node relies on the shared node state to determine which account to check.

Use this node, for example, in login flows where an account may already be created but not enabled until a later date.

Find more information in Account lockout for journeys.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `True`
- `False`

Properties

This node has no configurable properties.

Account Lockout node

Locks or unlocks the authenticating user's account profile.

For more information, refer to Account lockout for journeys.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

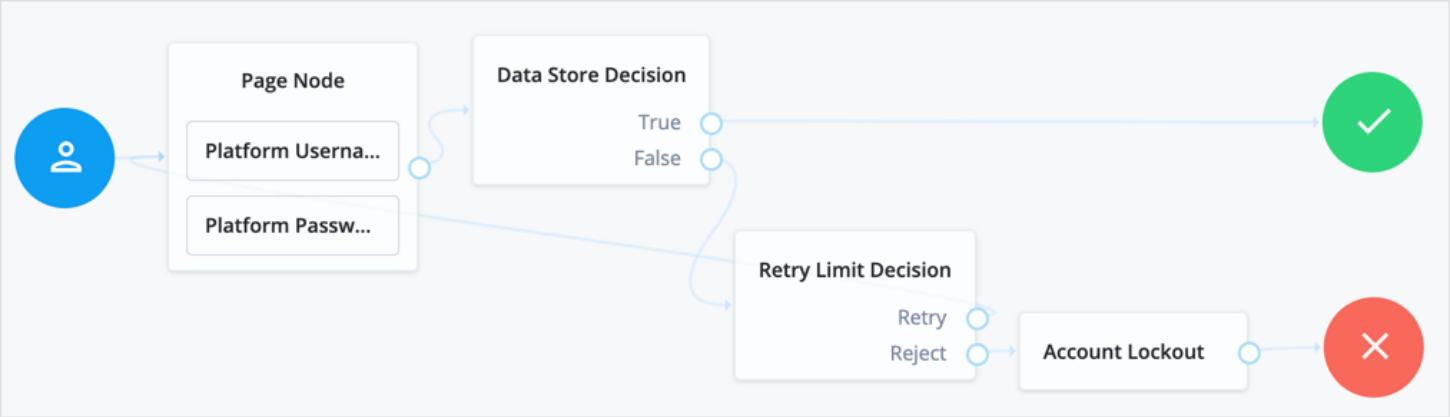
Single outcome path.

Properties

Property	Usage
Lock Action	Choose whether to `LOCK` or `UNLOCK` the authenticating user's account profile. The Data Store Decision node checks whether the account is locked.

Example

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



Auth Level Decision node

Compares the current authentication level value against a configured value.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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

Adds CAPTCHA support.

This node verifies the response token received from the CAPTCHA provider and creates a CAPTCHA callback for the UI to interact with.

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

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

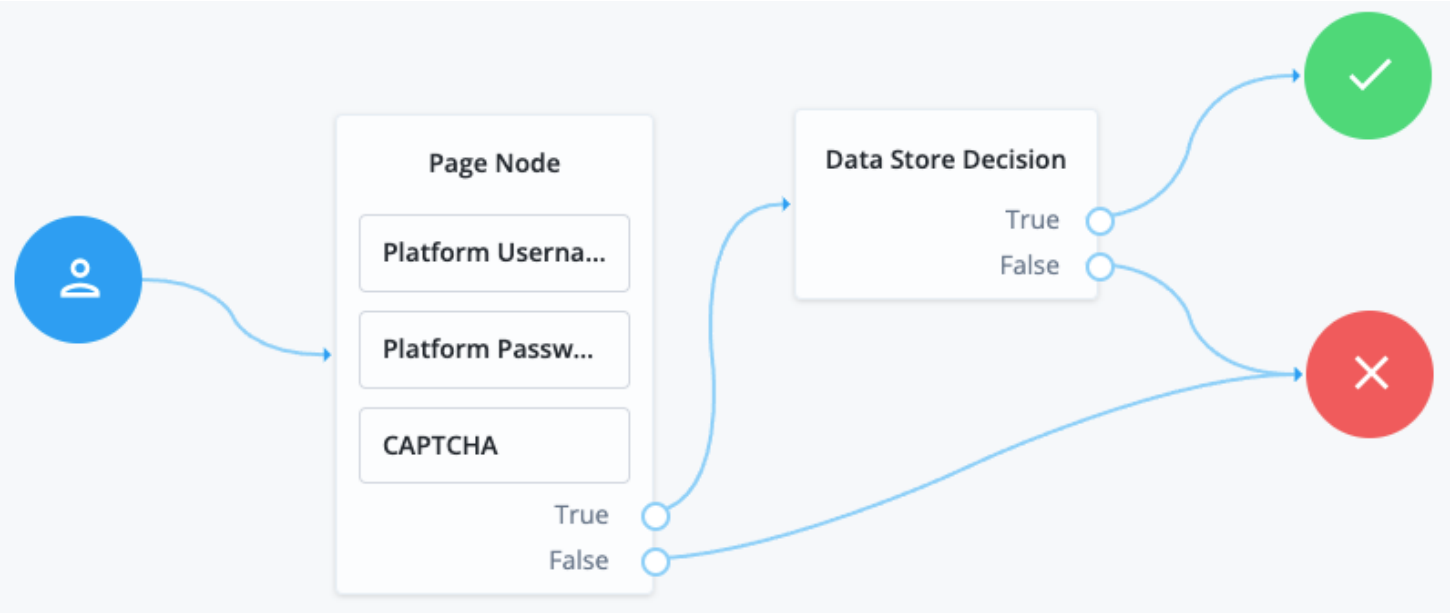
Outcomes

- `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`
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. For more information about score thresholds, refer to the Google documentation.

Example



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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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`
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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Behavioral nodes

Increment Login Count node

Increments the successful login count property of a managed object in IDM.

Use this node with the Login Count Decision node. To track the number of logins, include this node in the login authentication flows.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Properties

Property	Usage
Identity Attribute	The attribute used to identify the object in IDM.

Login Count Decision node

Triggers an action when a user's successful login count property reaches a specified number.

Add the Increment Login Count node to your login flows, so this node has the data to trigger a decision.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Properties

Property	Usage
Interval	The type of interval the decision should trigger on. To trigger the action once when the user reaches the number of successful login attempts, set **Interval** to `AT`. To trigger the action on every login attempt after the user reaches the number of successful login attempts, set **Interval** to `EVERY`.
Amount	The amount (count) of logins the interval should trigger on.
Identity Attribute	The attribute used to identify the object in IDM.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓

Product	Compatible?
Ping Identity Platform (self-managed)	✓

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

Validates a digital X.509 certificate collected by the Certificate Collector node.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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 could not validate the certificate. The node will use this path when it cannot validate the certificate, and no more specific outcome is available.

`Not found`

The **Match Certificate in LDAP** property is enabled, but the certificate was not 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.

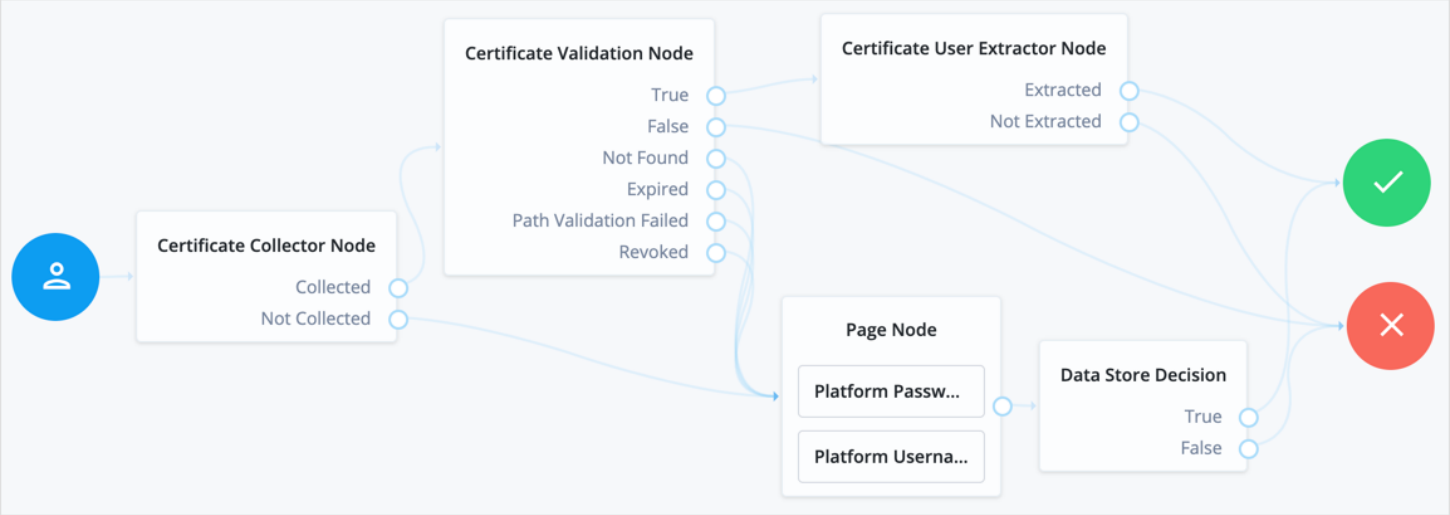
Properties

Property	Usage
Match Certificate in LDAP	When enabled, AM matches the certificate collected with the one stored in an LDAP directory entry. This entry and additional security-related properties are defined later in the node. Default: Disabled
Check Certificate Expiration	When enabled, AM checks whether the certificate has expired. Default: Disabled
Subject DN Attribute Used to Search LDAP for Certificates	Specifies the attribute that AM uses to search the LDAP directory for the certificate. The search filter also uses 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 CRL in the LDAP directory. Related properties are defined later in the node. Default: Disabled.
Issuer DN Attribute(s) Used to Search LDAP for CRLs	Specifies which attribute and value in the certificate Issuer DN AM uses to find the CRL in the LDAP directory. If only one attribute is specified, 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 the LDAP search filter used to find the CRL is `(CN=Some CA)`. Specify several CLRs for the same CA issuer in a comma-separated list (`,`) where the names are in the same order as they occur in the subject DN. In this case, the LDAP search filter used is `(cn=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=CN=Some CA,serialNumber=123456)`. Default: `CN`
HTTP Parameters for CRL Update	Specifies parameters that AM includes in any HTTP CRL call to the CA that issued the certificate. If the client or CA contains the Issuing Distribution Point Extension, AM uses this information to retrieve the CRL from the distribution point. Add the parameters as key pairs of values in a comma-separated list (`,`). For example, `param1=value1,param2=value2`.
Cache CRLs in Memory	(LDAP distribution points only) When enabled, AM caches CRLs. Default: Enabled

Property	Usage
Update CA CRLs from CRLDistributionPoint	When enabled, AM updates the CRLs stored in the LDAP directory store if the CA certificate includes either the `IssuingDistributionPoint` or the `CRLDistributionPoint` extensions. Default: Enabled
OCSP Validation	When enabled, AM checks the revocation status 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
LDAP Server Where Certificates are Stored	Specifies the LDAP server that holds the certificates. Enter each server in the `ldap-server:port` format. AM servers can be associated with LDAP servers by writing multiple chains with the format `am_server|ldapserver:port`. For example, `am.example.com|ldap1.example.com:636`. To configure a secure connection, enable the Use SSL/TLS for LDAP Access property.
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	Specifies the DN of the service account that AM uses to authenticate to the LDAP directory that holds the certificates. For example, `cn=LDAP User`. Default: `cn=Directory Manager`
LDAP Server Authentication Password	Specifies the password of the user configured in the **LDAP Server Authentication User** property.
Use SSL/TLS for LDAP Access	Specifies whether AM should use SSL/TLS to access the LDAP. When enabled, AM must be able to trust the LDAP server certificate. Default: Disabled

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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

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

Use this node with the Device Profile Collector node to determine if the authenticating user is on a previously saved, trusted device.

You can choose between two methods of comparison:

1. Built-in Matching

   The node handles the comparison and matching, and you can configure the acceptable variance, and specify a time frame that profiles are considered current.

2. Custom Matching

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

AM includes a template script you can customize to your requirements. In the AM admin UI, go to **Realms > *Realm Name* > Scripts**, and click **Device Match Template - Decision node Script**.

ForgeRock also provides a more complete sample script, as well as instructions for its use and a development toolkit. Find these resources on GitHub at https://github.com/ForgeRock/forgerock-device-match-script.

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

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

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

Evaluation continues along the `True` path if the collected device profile matches a saved profile, within the configured variance; otherwise, evaluation continues along the `False` path.

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

Properties

Property	Usage
Acceptable Variance	Specify the maximum amount of device attribute differences acceptable for a match.
Expiration	Specify the maximum age, in the number of days since being saved, that existing profiles can be considered for comparison. Device profiles saved to the user's profile before this time will not be compared to the collected metadata.

Property	Usage
Use Custom Matching Script	Specifies whether to use a custom script to compare the collected metadata with saved device profiles. The script type must be `Decision node script for authentication trees` (standalone AM) or `Journey Decision Node` (Ping Identity Platform deployment). **Note** When a custom matching script is used, the `Acceptable Variance` and `Expiration` properties are ignored. Default: `Authentication Tree Decision Node Script`
Custom Matching Script	Specifies the custom script to use if the **Use Custom Matching Script** property 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.

Device Profile Collector node

Gathers metadata about the device used to authenticate.

The node sends a `DeviceProfileCallback` callback. For more information, refer to Interactive callbacks.

When used with the Ping 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.
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 Access Management 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**.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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

Checks for the existence of a specified persistent cookie, the default being `session-jwt`.

If the cookie is present, the node verifies the signature of the JWT stored in the cookie with the signing key specified in the HMAC signing key property.

If the signature is valid, the node decrypts the payload of the JWT. It uses the key pair specified in the **Persistent Cookie Encryption Certificate Alias** property, found in the AM admin UI under **Realms** > ***Realm Name*** > **Authentication** > **Settings** > **Security**. The global level 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 identical.

> **Note**
>
> This node recreates the received persistent cookie, updating the value for the idle time property.
> Cookie creation properties for the Set Persistent Cookie node are therefore available in this node as well.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `True`
- `False`

Evaluation continues along the `True` outcome path if the persistent cookie is present and all the verification checks above are satisfied; otherwise, evaluation continues along the `False` outcome path.

Properties

Property	Usage
Idle Timeout	Specifies the maximum amount of 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)."

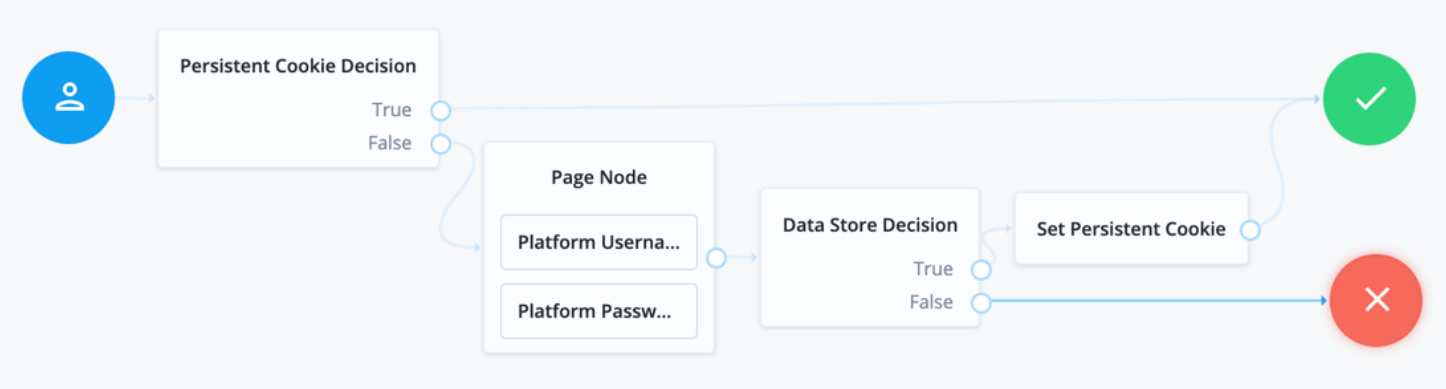
Property	Usage

HMAC Signing Key *(required)*	Specifies a key to use for HMAC signing of the persistent cookie. Values must be base64-encoded and at least 256 bits (32 bytes) long.
	
	Important
	To consume the persistent cookies generated by the Set Persistent Cookie node, ensure they are using the same HMAC signing key.
	
	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`

Persistent cookie name	Specifies the name of the persistent cookie to check.

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.

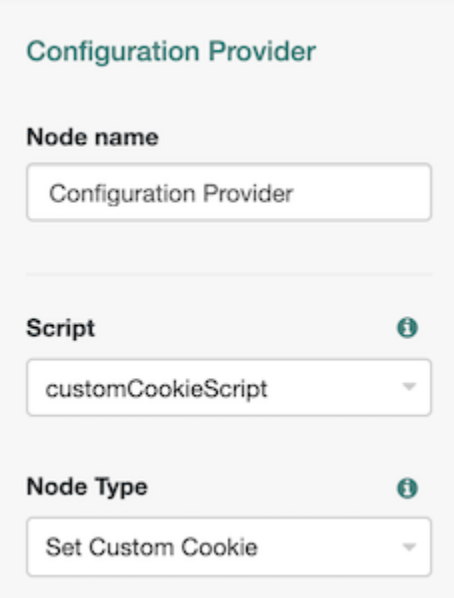
The node uses the specified properties to create a cookie with a custom name and value. It can also set attributes, such as the cookie path, domain, expiry, and security flags.

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`:



Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

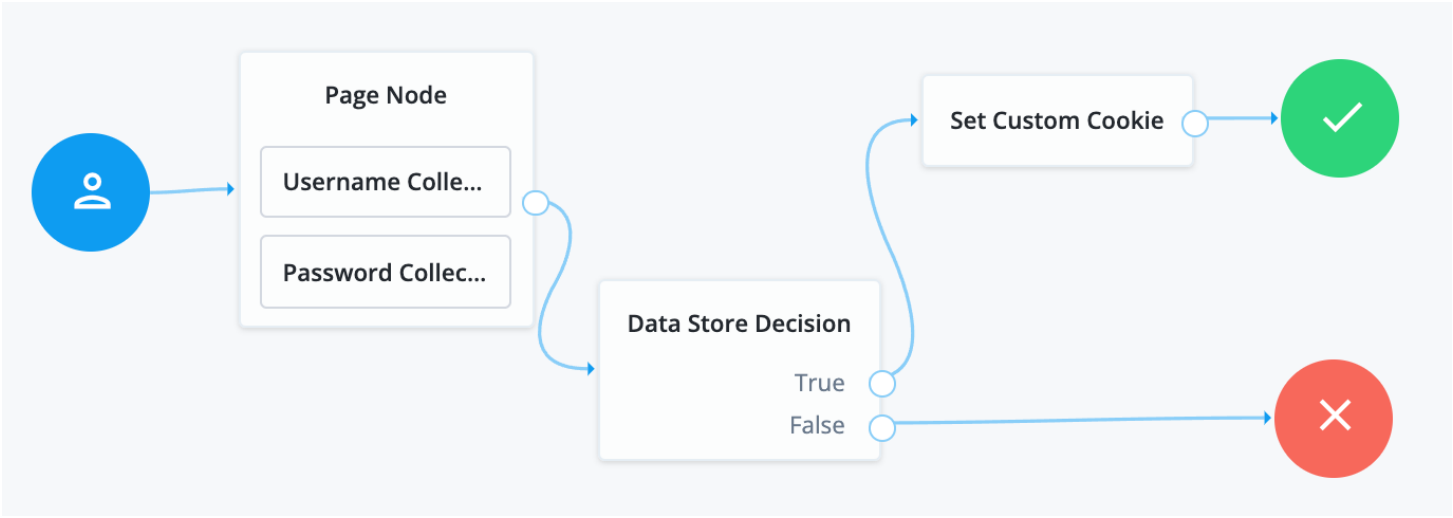
Outcomes

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. It uses the key pair specified in the **Persistent Cookie Encryption Certificate Alias** property, found in the AM admin UI under **Realms** > ***Realm Name*** > **Authentication** > **Settings** > **Security**. The global level is found under **Configure** > **Authentication** > **Core Attributes** > **Security**.

The node signs the cookie with the signing key specified in the HMAC signing key property. Any node that reads the persistent cookie must be configured with the same HMAC signing key.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

Property	Usage
Idle Timeout	Specifies 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	Specifies 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 *(required)*	Specifies a key to use for HMAC signing of the persistent cookie. Values must be base64-encoded and at least 256 bits (32 bytes) long.
	Important
	To consume the persistent cookies this node generates, ensure the nodes use the same HMAC signing key.
	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`

Persistent Cookie Name	Specifies the name used for the persistent cookie.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✕

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://openam.example.com:8443/openam/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. For more information, refer to 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✕

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://openam.example.com:8443/openam/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. For more information, refer to 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 an end user authenticate using an OIDC provider (OP)'s ID token. 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Inputs

None.

Dependencies

A valid OIDC ID token provided in the HTTP request header.

Properties

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 Id` is used instead. For example: `https://accounts.google.com/.well-known/openid-configuration`

Property	Usage
Client Secret Id	Specifies the ID of a client secret in the AM secret stores. Only required if the validation type is `Client Secret`.
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. For examples of transformation (normalization) scripts, refer to 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:

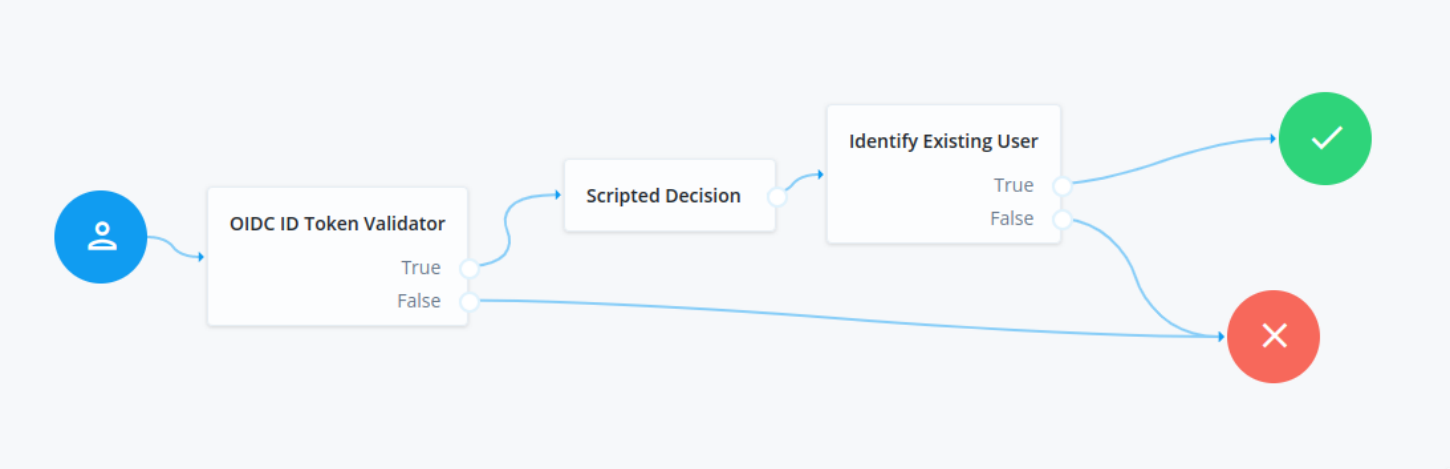
- `No OpenIdConnect ID Token referenced by header value: {}` : if the node can't read the ID token in the HTTP request header.

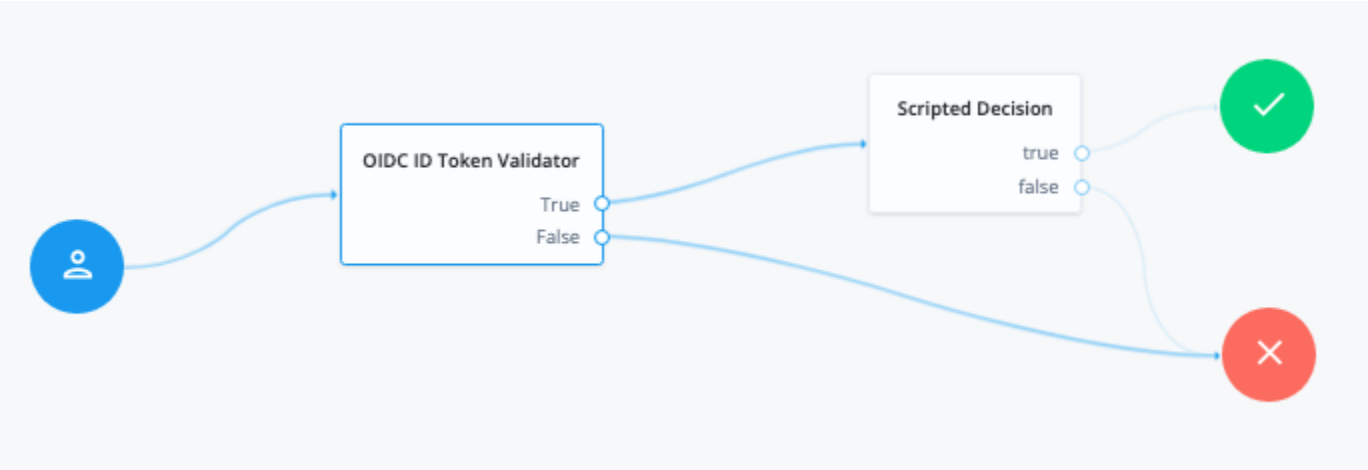
- `Error evaluating the script`: if there is a problem with the transformation script.

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.





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;
}());
```

In a Ping Identity Platform deployment, 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"
}
```

In a standalone AM deployment, 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✕

Outcomes

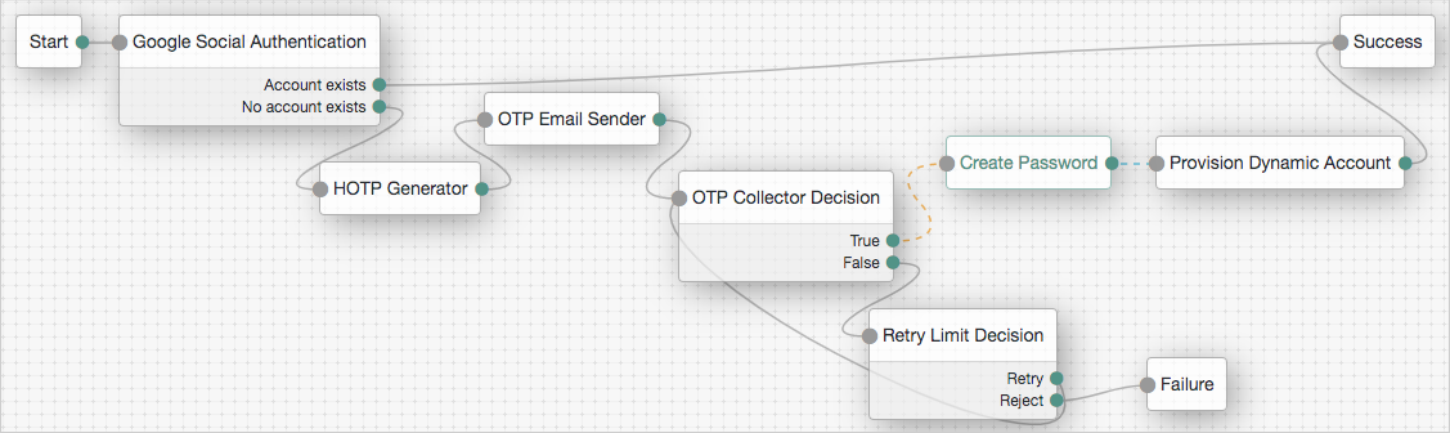
Single outcome path.

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 password 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**.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✕

Outcomes

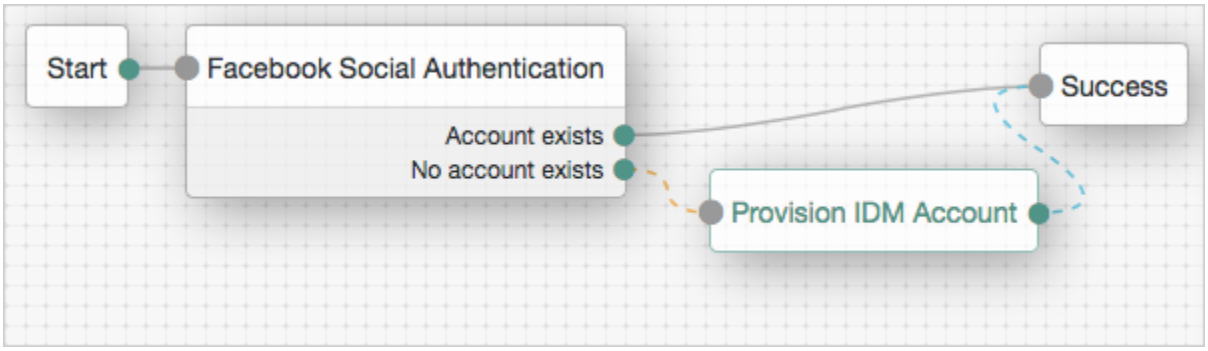
Single outcome path.

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.

  Refer to SSO and SLO in integrated mode.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

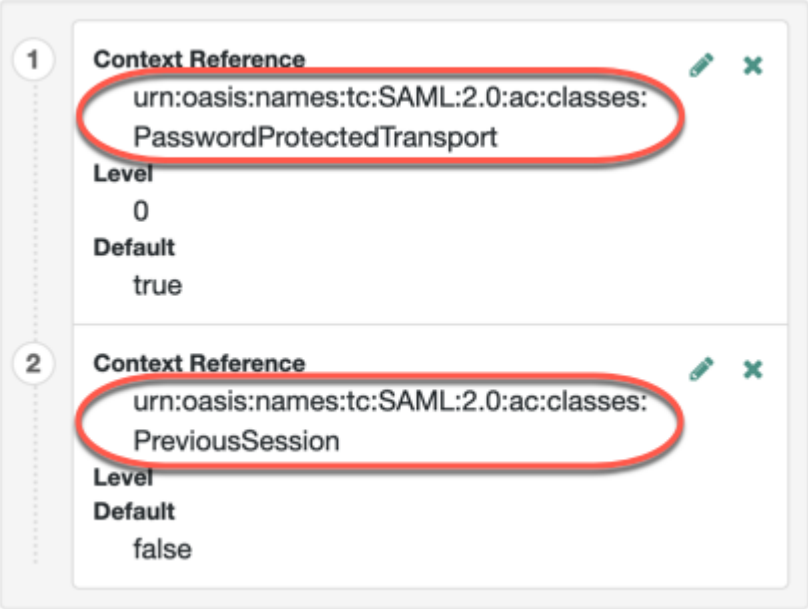
- `Account exists`
- `No account exists`

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`
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`

Property	Usage

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**.
	 
	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`

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`

Property	Usage
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`

For examples, refer to 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✕

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

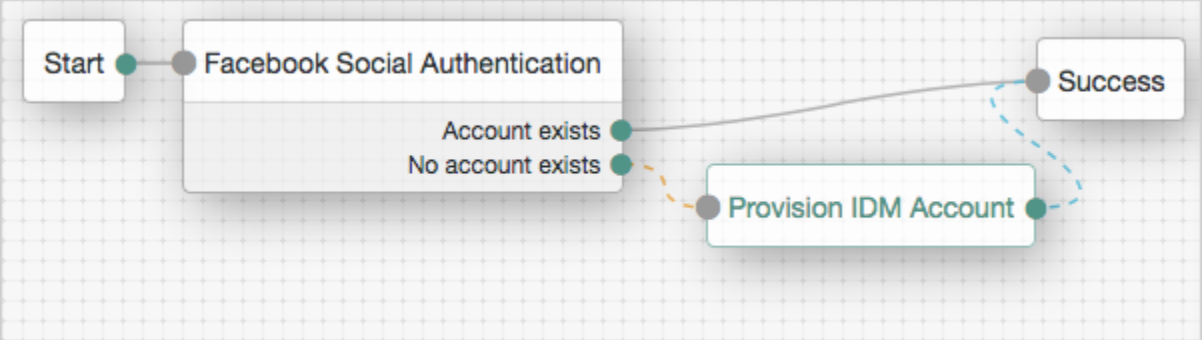
Property	Usage
Client ID	Specifies the `client_id` parameter as provided by Facebook.
Client Secret	Specifies the `client_secret` parameter as provided by Facebook.
Authentication Endpoint URL	Specifies the URL to the social provider's endpoint handling authentication as described in section 3.1 of The 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.
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://openam.example.com:8443/openam/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://openam.example.com:8443/openam/XUI/?realm=/mySubRealm`. For more information, refer to 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`

Property	Usage
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-`
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`.

Property	Usage
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`.
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. For more information, refer to 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✕

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.

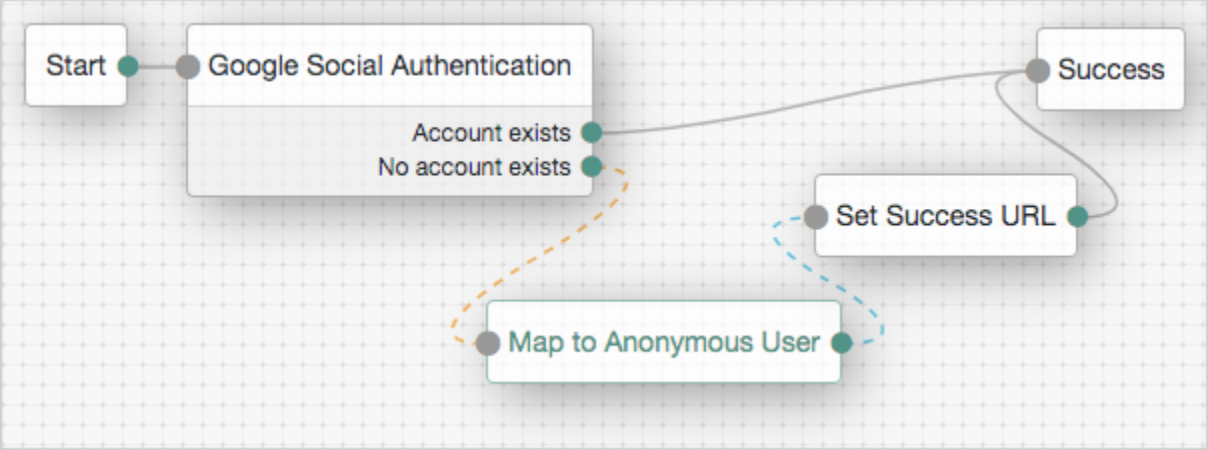
Property	Usage
Authentication Endpoint URL	Specifies the URL to the social provider's endpoint handling authentication as described in section 3.1 of The OAuth 2.0 Authorization Framework (RFC 6749). Default: `https://accounts.google.com/o/oauth2/v2/auth`
Access Token Endpoint URL	Specifies the URL to 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`.
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://openam.example.com:8443/openam/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://openam.example.com:8443/openam/XUI/?realm=/mySubRealm`. For more information, refer to 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`

Property	Usage
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-`
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`.

Property	Usage
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`.
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. For more information, refer to 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✕

Outcomes

Single outcome path.

Properties

This node has no configurable properties.

Social Provider Handler node

Takes the provider selection from the Select Identity Provider node and attempts to authenticate the user. This node collects relevant profile information from the provider and returns the user to the flow, transforming the profile information into the appropriate attributes.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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)*	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). In standalone AM deployments, select `Normalized Profile to Identity` or a custom script that transforms the profile to an identity object. To view the scripts and bindings, refer to normalized-profile-to-identity.js. 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. Normalization scripts (`<Identity provider>-profile-normalization.*`) are not suitable for this purpose.

Property	Usage
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.

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

To ensure existing users are dynamically linked, complete these additional steps:

In a standalone AM deployment:

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 a Ping Identity Platform deployment:

1. Connect the `No account exists` outcome to an Identify Existing User node.
2. Connect the Identify Existing User node to a Patch Object node to create the link.

`Social auth interrupted`

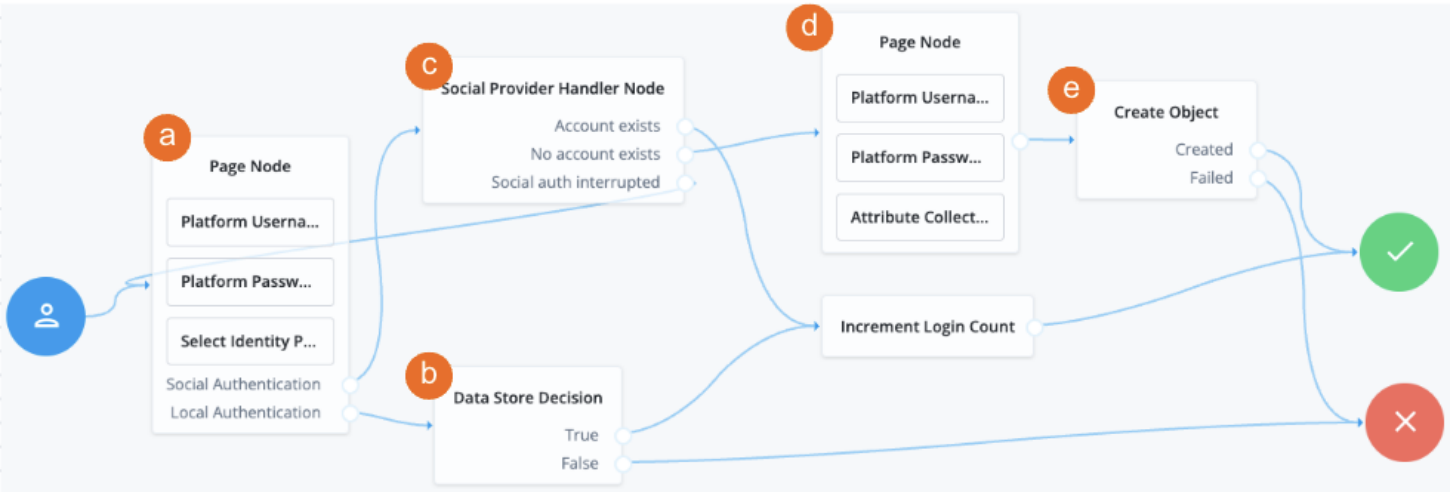
The user interrupted the social authentication journey after the node requested profile information from the social identity provider. This can happen in the following situations:

- 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.

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`.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

This node has no configurable properties.

For examples, refer to SSO and SLO in integrated mode.

Identity management nodes

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. Find more information in Terms and conditions.

Use this node for registration, or combined with the Terms and Conditions Decision node for progressive profiling or log in.

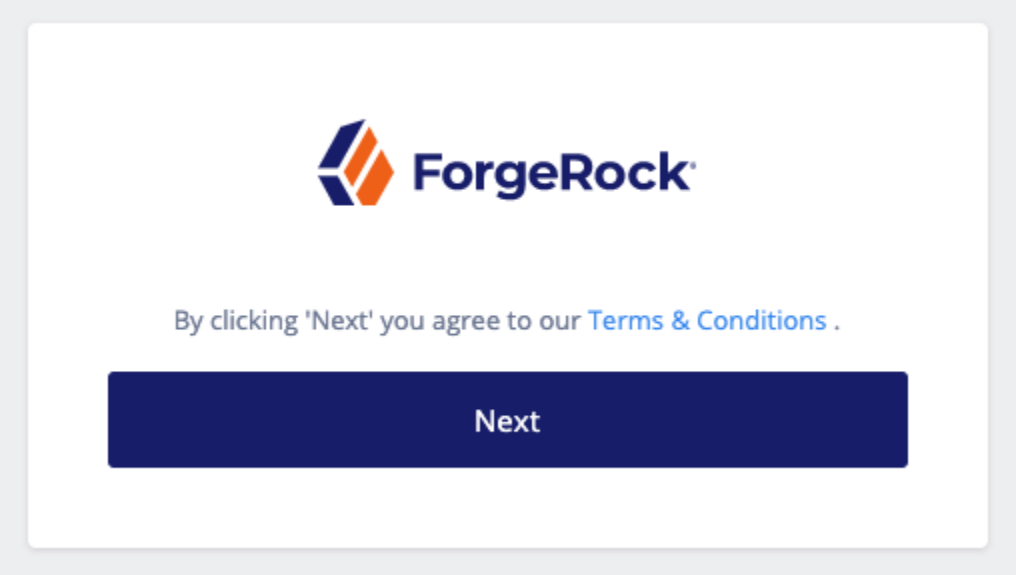
Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

The user must accept the terms and conditions in order to proceed.



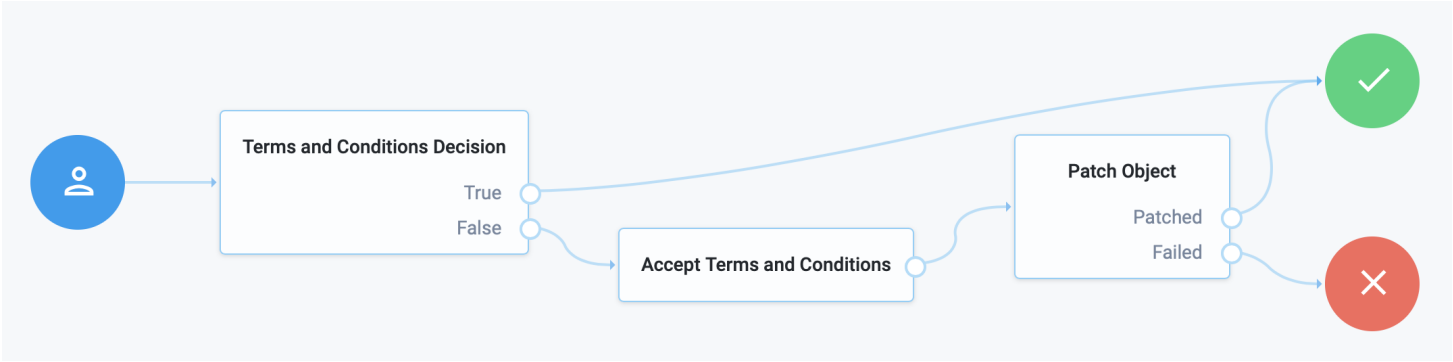
Properties

This node has no configurable properties.

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.

If the user accepts, the acceptance response is stored in IDM:



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 validate them through IDM's policy filter.

Use the node alone or within a Page node.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

Property	Usage
Attributes to Collect	A list of the attributes to collect based on those in the IDM schema for the current identity object.
All Attributes Required	When enabled, all attributes collected in this node are required in order to continue.

Property	Usage
Validate Input	When enabled, validate the content against any policies specified in the IDM schema for each collected attribute. Find more information in Use policies to validate data in the IDM documentation. If you enable this property, the collected attributes must be *User Editable* in IDM. 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. For details, refer to Property Configuration Properties in the IDM documentation.
Identity Attribute	The attribute used to identify the object in IDM.

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 cannot return the value of the attribute, but can work with private attributes.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `True`

- `False`

Properties

Property	Usage
Present Attribute	The object attribute to verify is present in the IDM object. This can be an otherwise private attribute, such as `password`.
Identity Attribute	The attribute used to identify the object in IDM.

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`.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

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 object in IDM.

Consent Collector node

Prompts the user for consent to share their profile data.

A consent notice is listed for each IDM mapping that has consent enabled. If an IDM 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

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

Creates 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Properties

Property	Usage
Identity Resource	The type of IDM 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.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✕

Outcomes

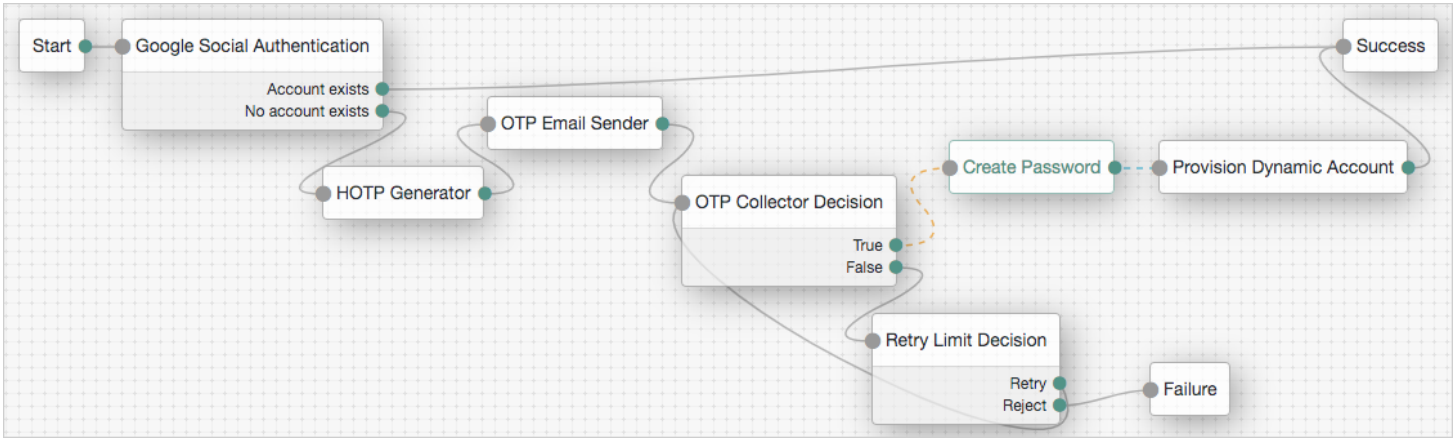
Single outcome path.

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 password 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓

Product	Compatible?
Ping Identity Platform (self-managed)	✓

Properties

Property	Usage
User Name	The attribute used to identify the username in an IDM object.
Identity Attribute	The attribute used to identify the 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

Verifies 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.

Compatibility

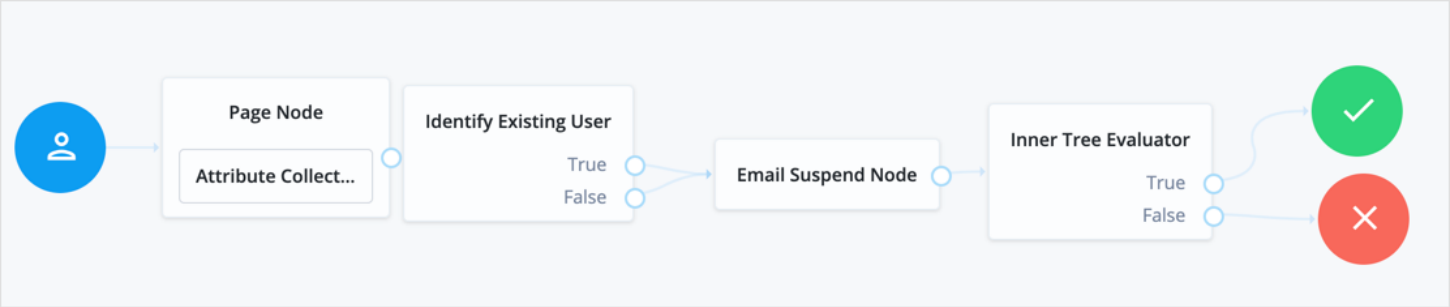
Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Properties

Property	Usage
Identifier	The attribute to collect from an IDM object.
Identity Attribute	The attribute used to identify the 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.

Example

The following is an example of a forgotten password flow. The user enters information that the Identify Existing User node uses to try to identify them. Next, AM uses the Email Suspend node to send an email to the user and suspend authentication. Once authentication resumes, AM 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

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 object in IDM.

KBA Definition node

Collects KBA questions and answers and saves them to the user profile.

Use this node when creating or updating a user with Knowledge-Based Authentication enabled. For more information, refer to Security questions.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Properties

Property	Usage
Purpose Message	A localized message describing the purpose of the data requested from the user.

Property	Usage
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**.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Properties

Property	Usage
KBA Attribute	The IDM object attribute in which KBA questions and answers are stored.
Identity Attribute	The attribute used to identify the 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.

  For details, refer to the OpenICF documentation.

- If you plan to collect credentials in the identity repository for users, synchronize accounts from the third-party service.

  For details, refer to Synchronization in the IDM documentation.

Use this node after collecting the authentication credentials. For example, use the Username Collector node and the Password Collector node (standalone AM) or the Platform Username node and the Platform Password node (Ping Identity Platform deployment) to collect the username and password.

Pass the credentials to this node to authenticate the identity against the service.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

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

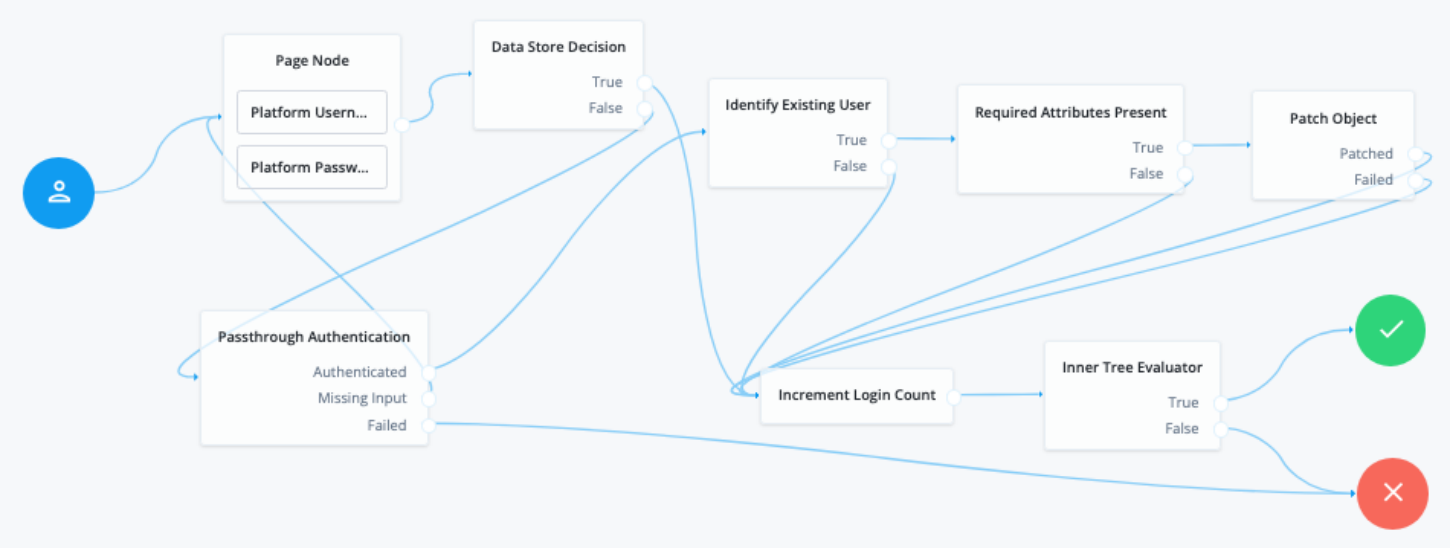
Property	Usage
System Endpoint	Required. Name of the connector to the third-party service that performs authentication.
Object Type	The OpenICF object type for the object being authenticated. Default: `account`
Identity Attribute	The username attribute for authentication. Default: `userName`
Password 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

Source node	Outcome path	Target node
	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

Patches the attributes in an existing managed object in IDM.

Use this node for progressive profile completion to collect additional profile data from a user after they have logged in several times.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Properties

Property	Usage
Patch as Object	Allows patching as the object being updated. Enable this property to patch a user object as part of the user's current session, for example, when updating their password.

Property	Usage
Ignored Fields	Fields from the shared node state that should be ignored as part of patch. Use this to patch only the fields you want to update. If this is empty, the node attempts to update all the node shared state fields as part of the patch.
Identity Resource	The type of IDM 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.
Identity Attribute	The attribute used to identify the object to update in IDM.

Platform Password node

Prompts the user to enter their password and stores the input in a configurable state attribute.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

Property	Usage
Validate Password	When enabled, this node checks the user's input against IDM's password policies, and 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\" }" ] }`
Password Attribute	The attribute used to store a password in the IDM 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

Platform Username node

Prompts the user to enter their username and stores it in a configurable state attribute.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓

Product	Compatible?
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

Property	Usage
Validate Username	When enabled, this node checks the user's input against IDM's username policies, and 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.
Username Attribute	The attribute used to store a username in the IDM object.

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`.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓

Product	Compatible?
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

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 object in IDM.

Query Filter Decision node

Checks if the contents of a user's profile matches a specified query filter.

Use this node to verify whether a particular field has been filled, or that the contents of a field match a specific pattern. For instance, use this in progressive profile flows to check if marketing preferences are set on a user's profile.

For more information on constructing effective query filters, refer to Construct queries in the IDM documentation.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓

Product	Compatible?
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `True`
- `False`

Properties

Property	Usage
Query Filter	A query filter used to check the contents of an object.
Identity Attribute	The attribute used to identify the object queried in IDM.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `True`
- `False`

Properties

Property	Usage
Identity Resource	The type of IDM 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.

In a platform deployment, 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 (platform deployment), 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓

Product	Compatible?
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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. For more information, refer to 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `True`
- `False`

Properties

Property	Usage
Identity Attribute	The attribute used to identify the object to check 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

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 object to update in IDM.

Utility nodes

Agent Data Store Decision node

Verifies that a provided agent ID and password match a web agent or Java agent profile configured in AM. Obtain the web or Java agent ID and password with a Zero Page Login Collector node.

> **Note**
>
> Non-agent identities, such as users stored in configured identity repositories, cannot be verified by using this node. Use the Data Store Decision node instead.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `True`
- `False`

Evaluation continues along the `True` path if the credentials match those of a configured agent profile; otherwise, the evaluation continues along the `False` path.

Properties

This node has no configurable properties.

Anonymous Session Upgrade node

Upgrades an anonymous session to a non-anonymous session.

Use this as the first node in the flow.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

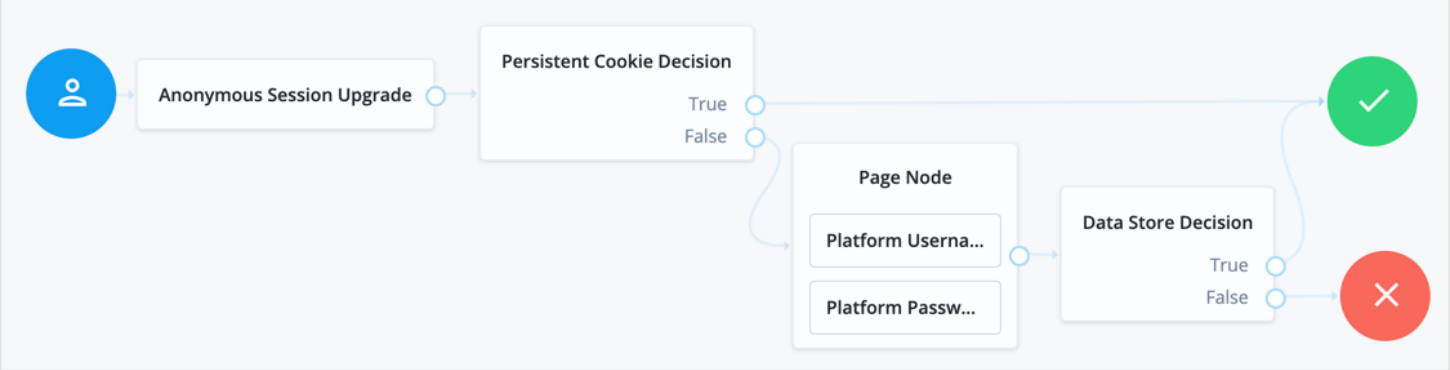
Single outcome path.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

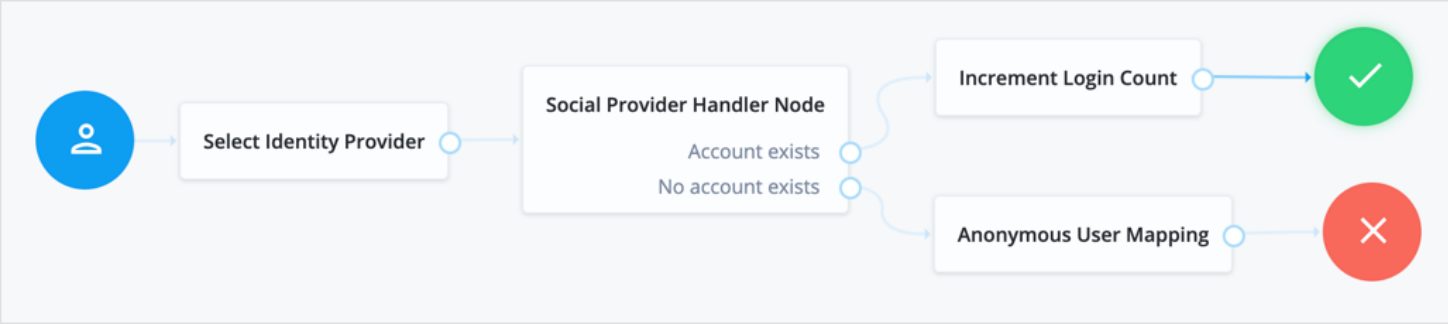
Single outcome path.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Field Display Type	Specifies the format of the options presented to the user.
	Note
	This property only appears when the node is placed within a Page node.
	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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

   In the Ping Identity Platform admin UI for a Ping Identity Platform deployment:

   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`

In AM admin UI for a standalone AM deployment, 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": {}
}
```

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

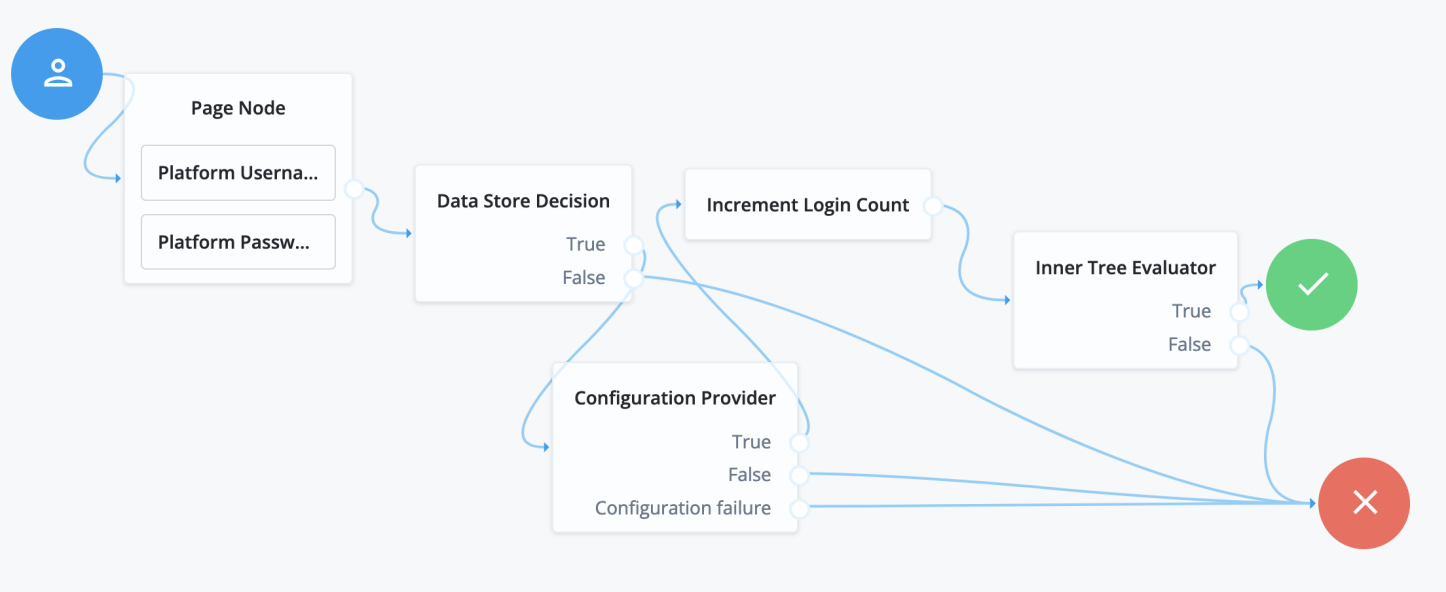
`Message Node`

Script Inputs

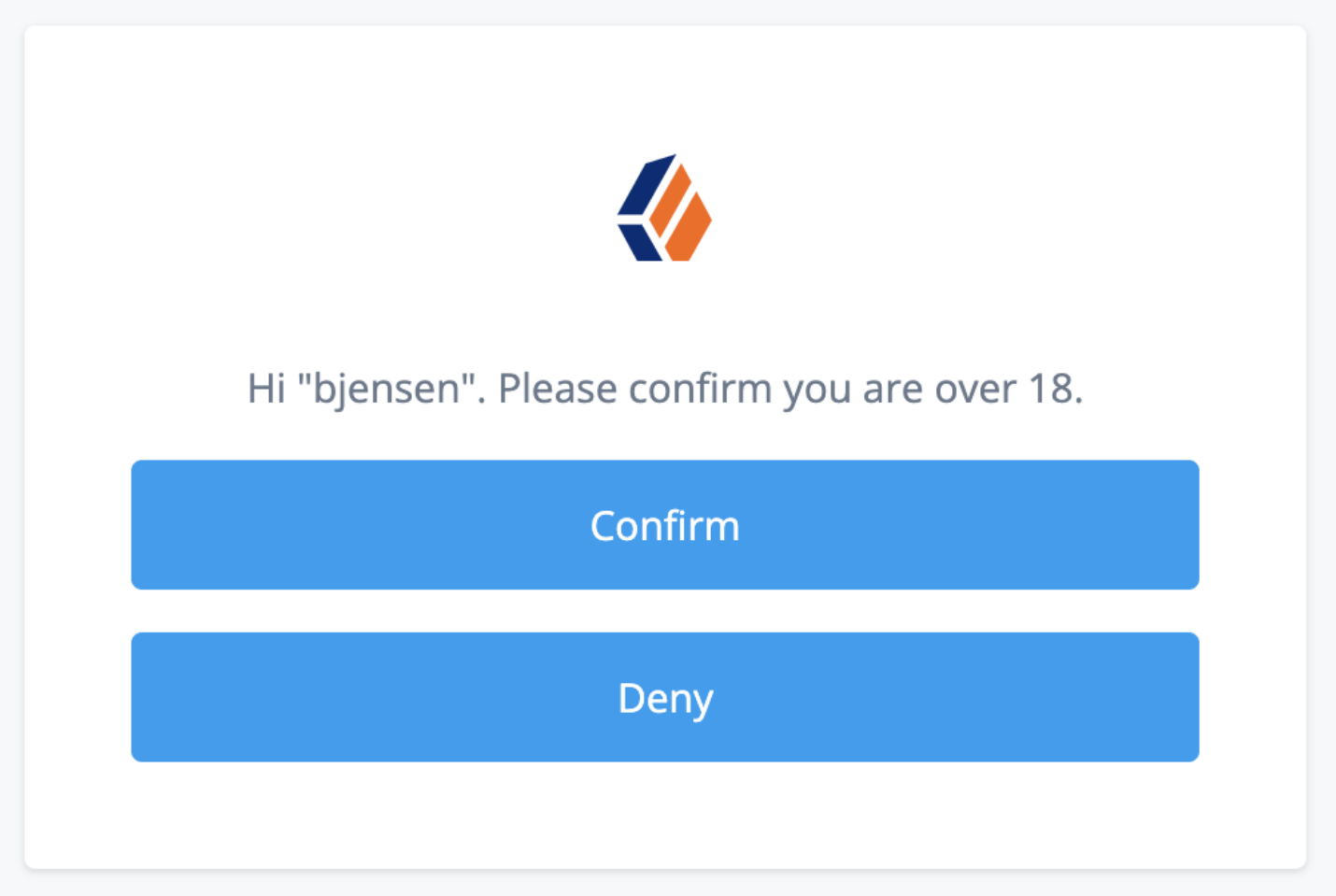
`username`

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:



- 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. It 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`.

Find more information 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

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 IDM property to match in the managed identity profile 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`

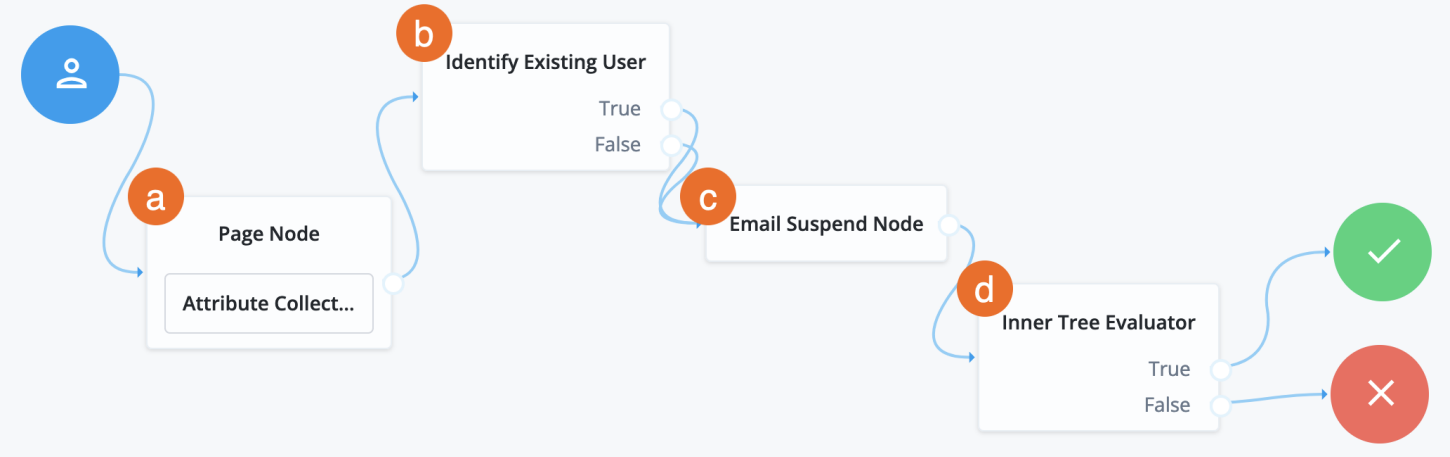
Forgotten username

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

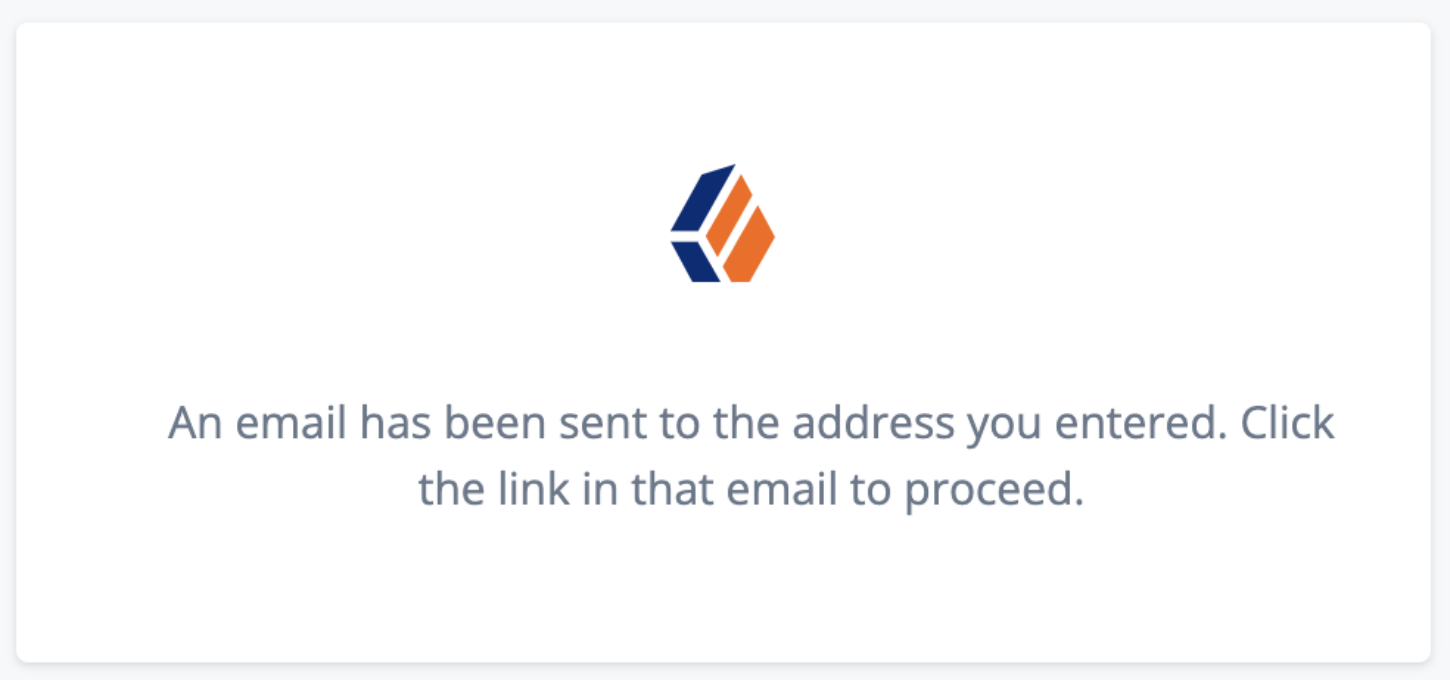


a The Page node with an Attribute Collector node prompts for the end user's email address.

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:



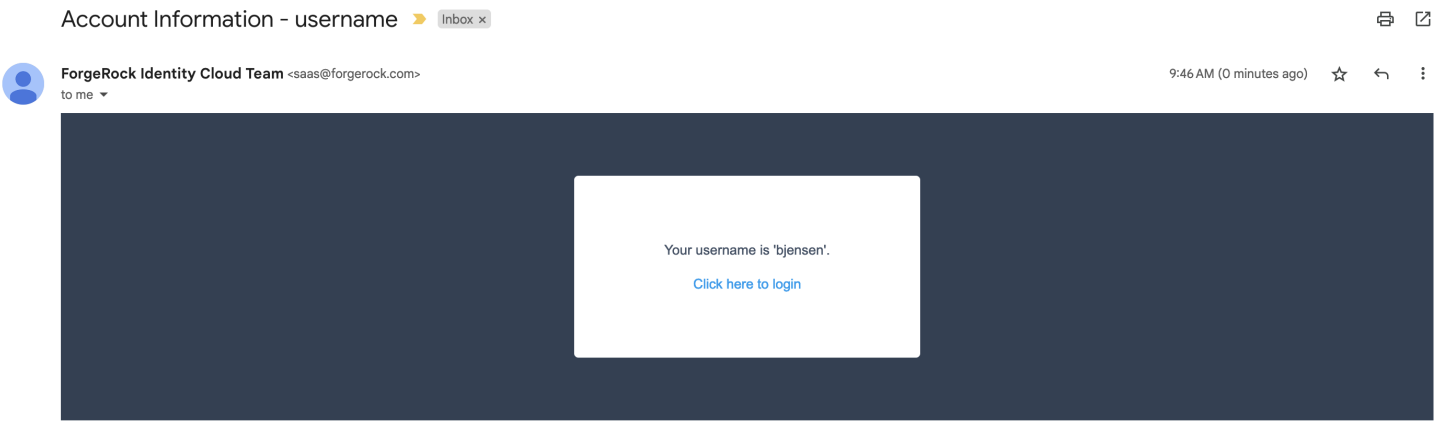
The node's settings are:

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:



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. It relies on the email service configured in IDM to send an 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed) **Note** This functionality requires that you configure AM as part of a Ping Identity Platform deployment.	✓
Ping Identity Platform (self-managed)	✓

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 IDM property to match in the managed identity profile. 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

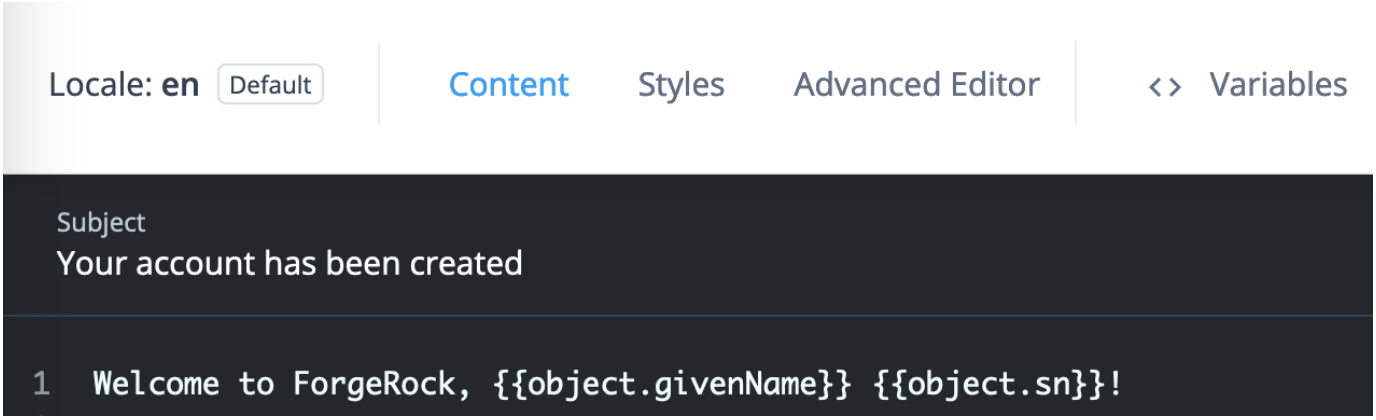
Use the **Email Template** node to send an email message when the journey doesn't depend on a reply. For example, send a welcome message when a user completes registration.

This example augments the default **Registration** journey and sends a welcome email.

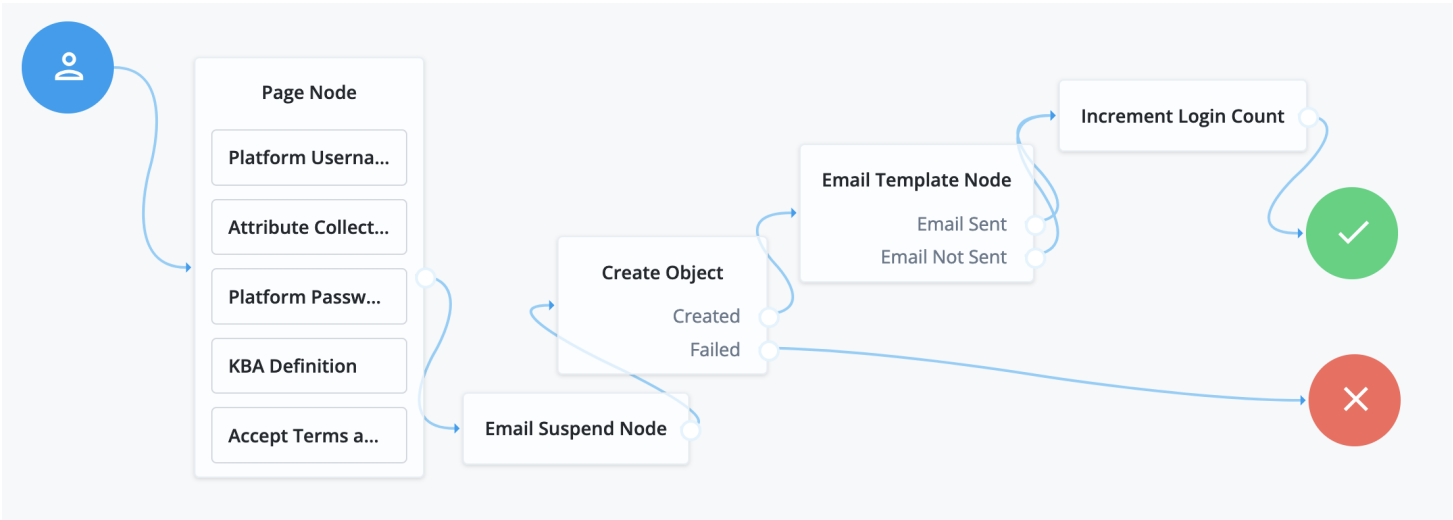
Before setting up the journey:

- 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:



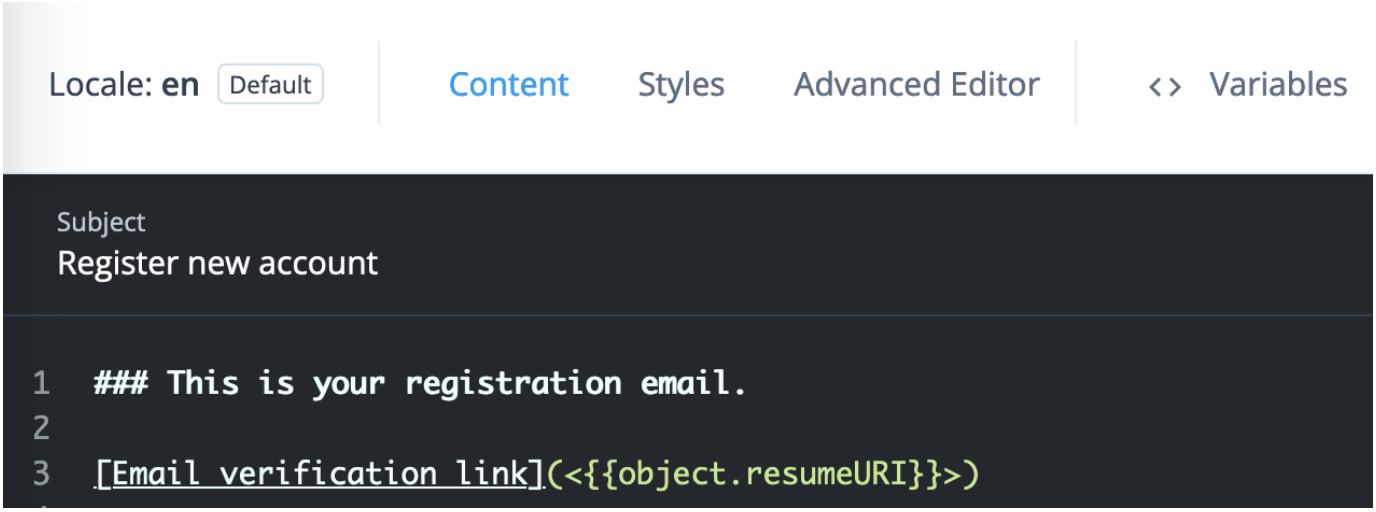
The journey is as follows:



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:



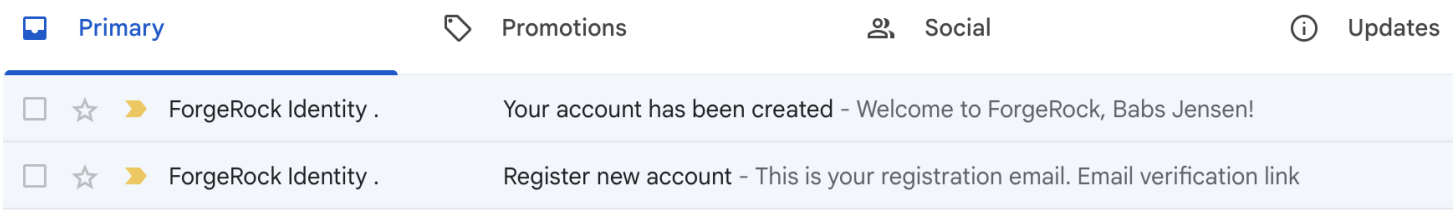
3. The Create Object node stores the newly registered user's profile.
4. The **Email Template** node reads the user profile and populates the template from profile attribute values. It makes a request to the email service to send the message.

   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:



- 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.

For more information on how AM determines the redirection URL, and to configure the Validation Service to trust redirection URLs, refer to Configure success and failure redirection URLs.

> **Tip**
>
> The URL is also saved in the shared `nodeState` object on the `failureUrl` key.
> For more information, refer to Customize authentication trees.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

Property	Usage
Failure URL *(required)*	Specify the full URL to redirect to when authentication fails.

Get Session Data node

Retrieves the value of a specified key from a user's session data, and stores it in the specified key in the shared `nodeState` object.

This node is only used during session upgrade—when the user has already successfully authenticated previously—and is now upgrading their session for additional access. Find more information on upgrading a session in Session upgrade.

This node fails with an error if you attempt to get a property when the user does not have an existing session. Use a Scripted Decision node with a script that determines if an existing session is present:

```
if (typeof existingSession !== 'undefined') {
  outcome = "hasSession";
} else {
  outcome = "noSession";
}
```

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

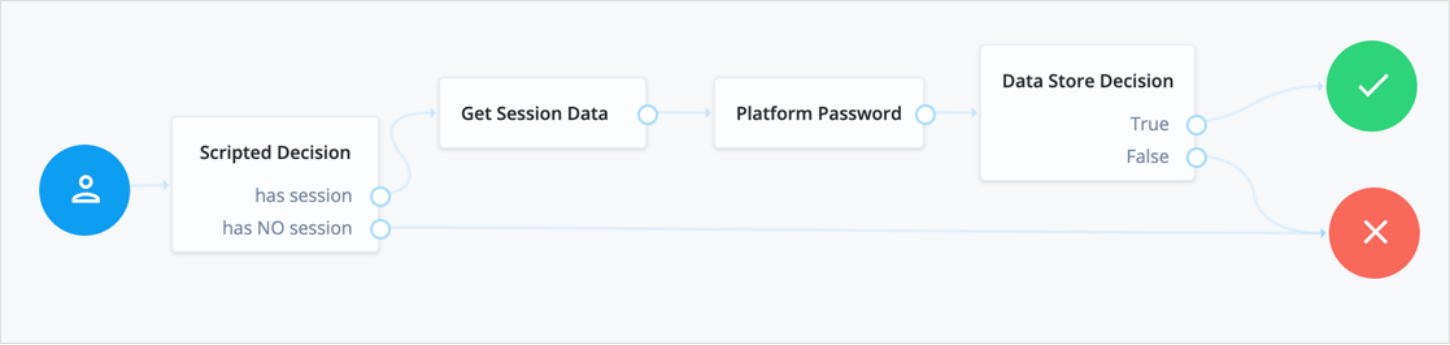
Outcomes

Single outcome path.

Properties

Property	Usage
Session Data Key *(required)*	Specify the name of a key in the user's session data used to retrieve the value.
Shared State Key *(required)*	Specify the name of a key in the `nodeState` object used to store the retrieved value.

Example



The following table includes example keys that may be available in an existing session and the 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`
`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

Lets you nest authentication journeys as children within a parent. There is no limit to the depth of nesting.

Any information collected or set by the parent journey, such as a username or the authentication level, is available in the child journeys.

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.

Find more information on shared state data in Access shared state data.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

- `True`
- `False`

Evaluation continues along the `True` path if the child reached the `Success` exit point; otherwise, evaluation continues along the `False` path.

Properties

Property	Usage
Tree name *(required)*	Enter the name of the authentication journey to evaluate.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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**. 1. 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 [.label]#Pencil# icon (✎). To remove an entry, click the [.label]#Delete# icon (🗑).
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 [.label]#Pencil# icon (✎). To remove an entry, click the [.label]#Delete# icon (🗑).

Property	Usage
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 [.label]#Pencil# icon (✎). To remove an entry, click the [.label]#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	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.
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.

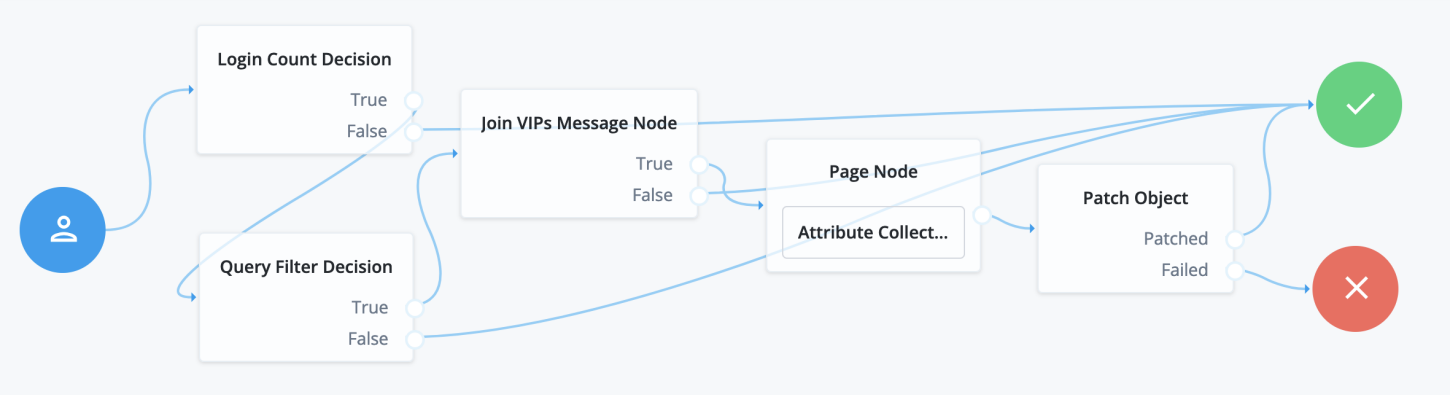
If this node encounters an internal configuration issue, it logs a warning message `Error attempting to retrieve the realm/global default locale`. If the warning persists, contact Ping Identity Support.

Examples

Use a **Message** node to:

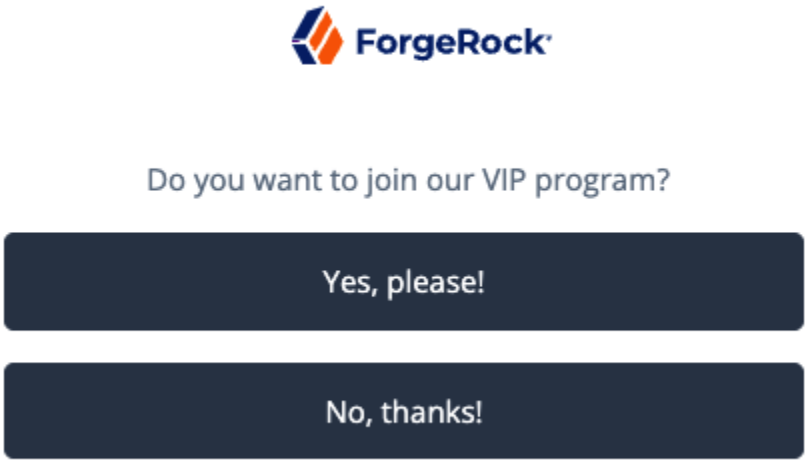
- Communicate an important message for the user to acknowledge.
- Ask a question with a yes/no answer.

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

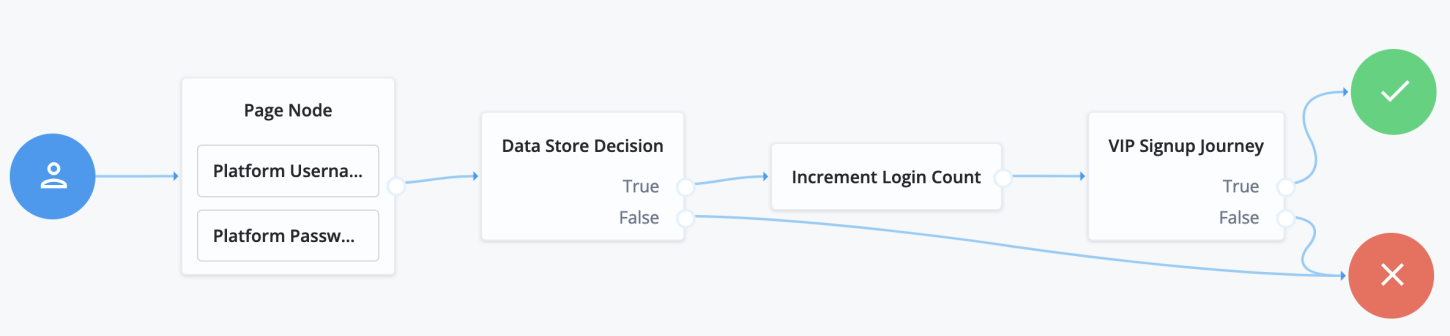
`en-gb` ; `Yes, please!`

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.

For information on the `Meter` metric type, refer to Monitoring metric types. The metric is exposed in all available interfaces, as described in Monitor AM instances.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

Property	Usage
Metric Key *(required)*	Specify the name of a metric to increment when evaluation passes through the node.

Page node

Combines multiple nodes that request input into a single page for display to the user.

Drag and drop nodes on to the page node to combine them. Only add nodes that use callbacks to request input. Do not add other nodes, such as the Data Store Decision node and the Push Sender node to this node.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Properties

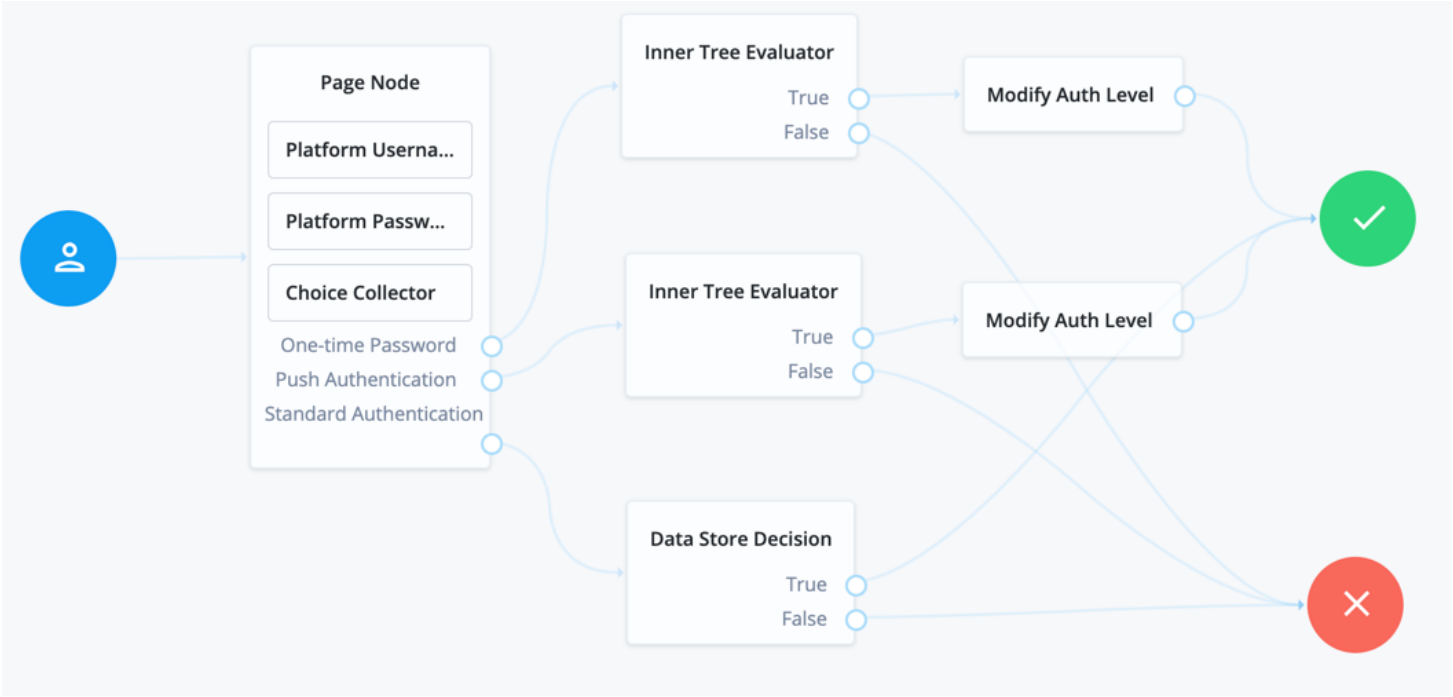
Property	Usage
Page Header	Optional localized title for the page node and the nodes contained within it. Use this when components of an authentication flow need a title, such as breaking a registration into labeled sections.
Page Description	Optional localized description for the page node and the nodes contained within it. Use this when additional descriptive text is needed in an authentication flow.
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.

Property	Usage
Page Footer	Optional. A localized footer for the page node and the nodes contained within it. Use this when additional descriptive text is needed in an authentication flow. 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.

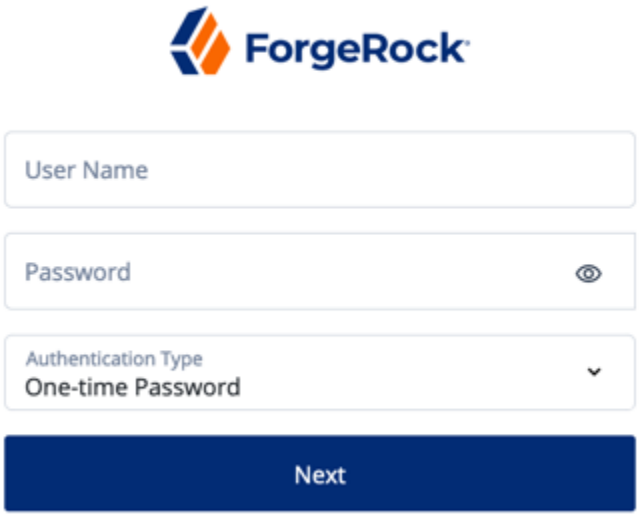
> **Note**
>
> This node's optional properties are passed in the response, but a self-hosted or custom UI must support these properties to make them visible to the end user.

Example

The following example uses a page node containing a username collector, a password collector, and a choice collector:



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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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`
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-US`. For information on valid locale strings, refer to JDK 11 Supported Locales. 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. For information about customizing and translating the default messages, refer to Internationalization.
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	Specifies 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-US`. For information on valid locale strings, refer to JDK 11 Supported Locales. 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. For information about customizing and translating the default messages, refer to Internationalization.

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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 [.label]#Pencil# icon (🖉). To remove an entry, click the [.label]#Delete# icon (🗑).

Property	Usage
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 [.label]#Pencil# icon (✎). To remove an entry, click the [.label]#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**.

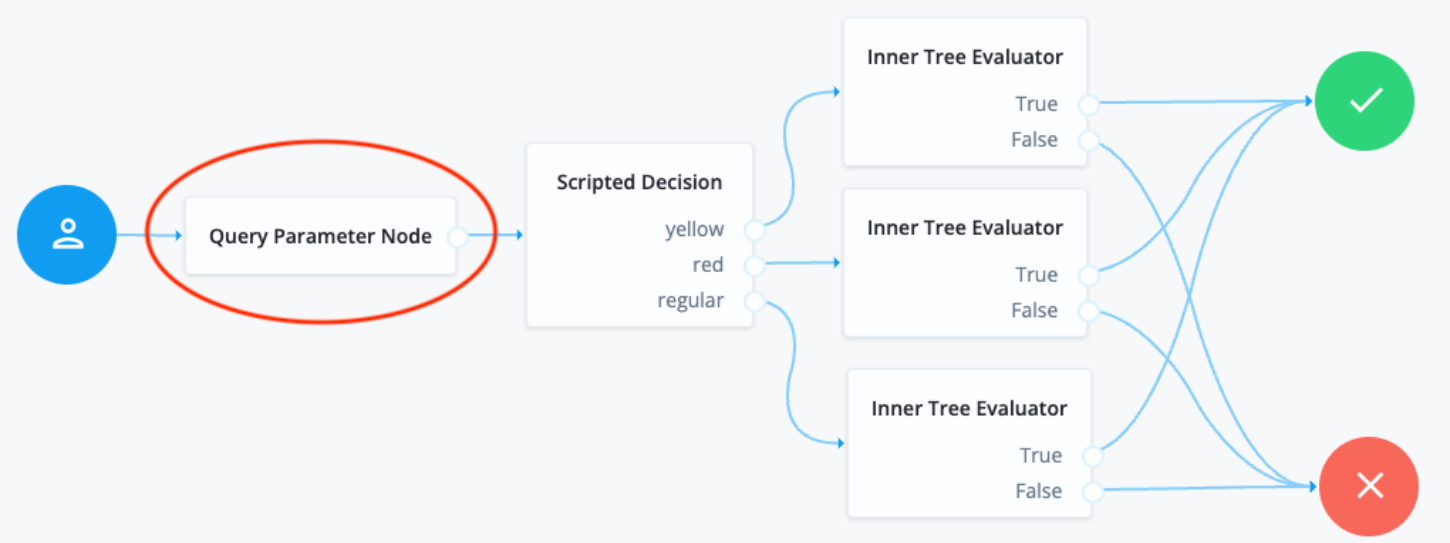
Examples

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:

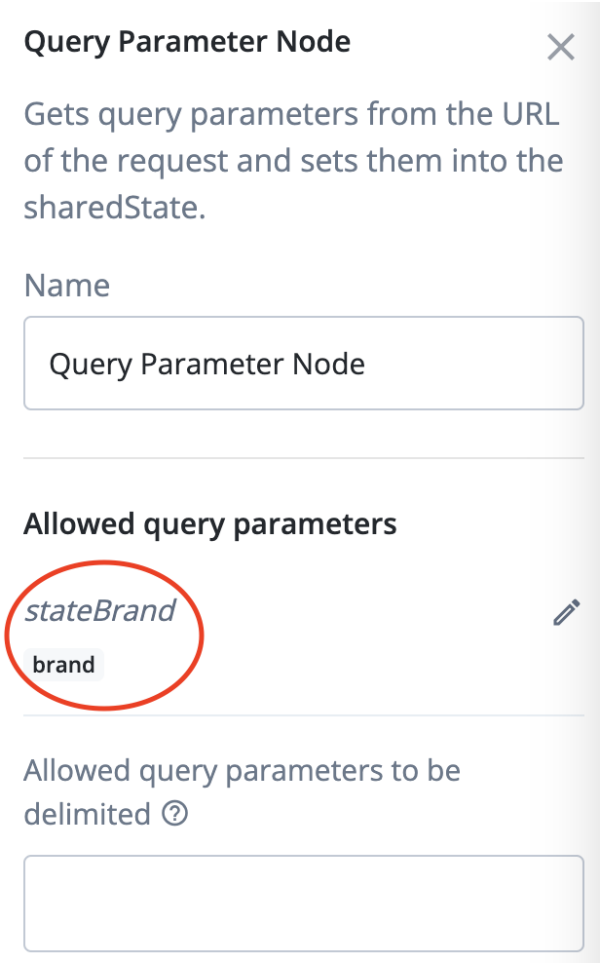
Customized branding

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`

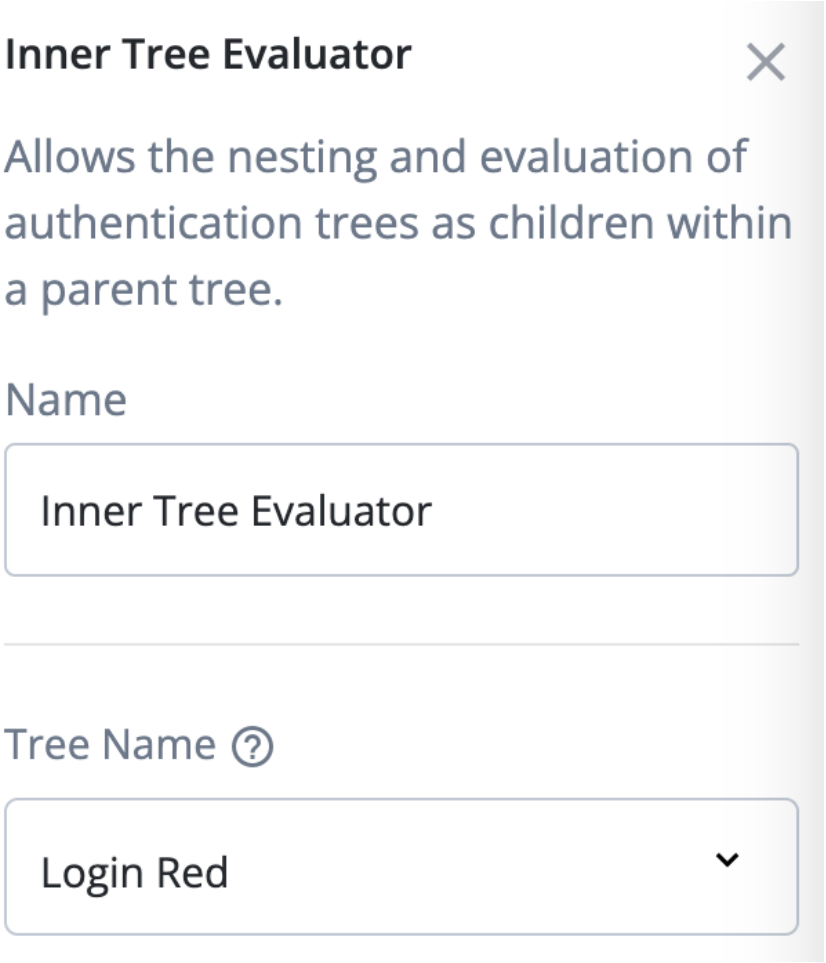


2. The user accesses the journey at a URL that can be one of the following:
   - `https://<tenant-env-fqdn>/am/XUI/?realm=alpha&authIndexType=service&authIndexValue=Login`
   - `https://<tenant-env-fqdn>/am/XUI/?realm=alpha&authIndexType=service&authIndexValue=Login&brand=yellow`
   - `https://<tenant-env-fqdn>/am/XUI/?realm=alpha&authIndexType=service&authIndexValue=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:



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.

For more information on webhooks, refer to Configure authentication webhooks.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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`.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

Property	Usage
Property Names *(required)*	Enter one or more key names of properties to remove from the session.

Retry Limit Decision node

Permits the specified number of passes through to the `Retry` outcome path before continuing evaluation along the `Reject` outcome path.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

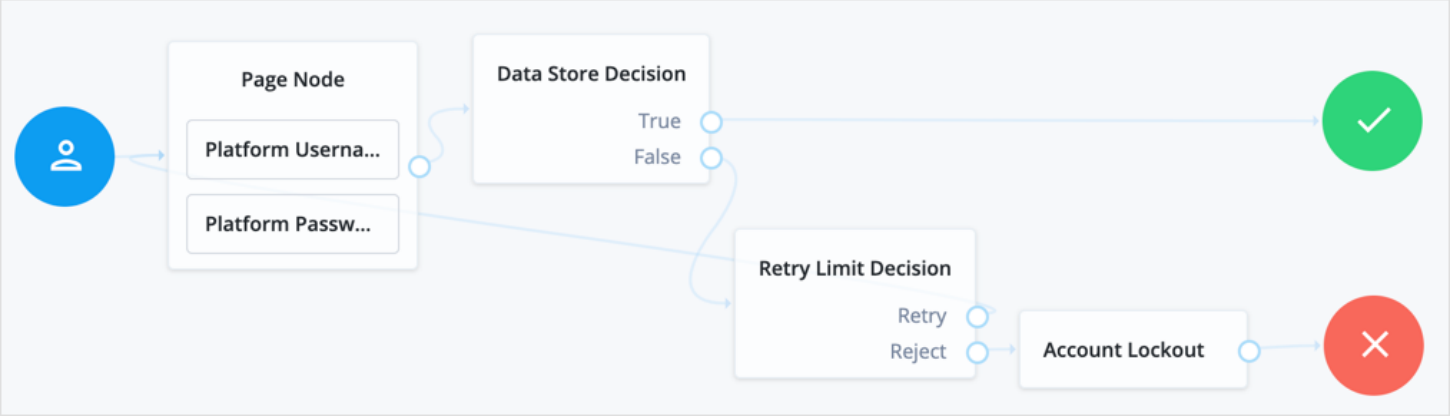
- `Retry`
- `Reject`

Properties

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 between successful authentications. Possible values are: ***Enabled*** The node saves the number of failed login attempts to 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 cannot find the user's profile, authentication ends with an error. ***Disabled*** The node saves the number of failed login attempt in the `nodeRetryLimitKey` shared state property, which is discarded when the authentication session ends. For security reasons, ForgeRock recommends that you enable this setting. Default: Enabled.

Example



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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓

Product	Compatible?
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

For details about the inputs available to the script, refer to 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

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

script. The script gets the user's names from their profile and stores a message in a shared state property:

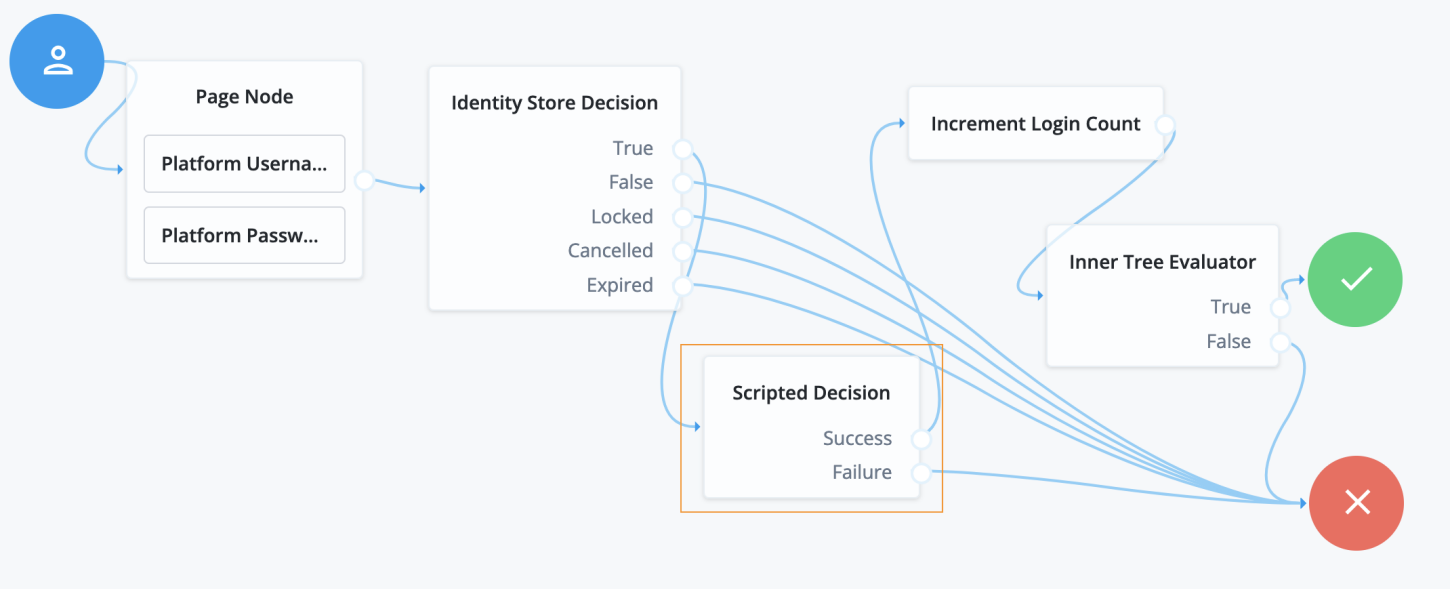
```
var goTo = org.forgerock.openam.auth.node.api.Action.goTo

// 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

Add `key:value` properties to the user's session if authentication is successful.

> **Tip**
>
> You can access session properties using a variable in a webhook. For more information, refer to Configure authentication webhooks.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Evaluation continues after setting the specified properties in the session.

Properties

Property	Usage
Properties	To add a session property, click **Add**, enter a key name and a value, and then click +. Repeat the steps to add multiple properties.

State Metadata node

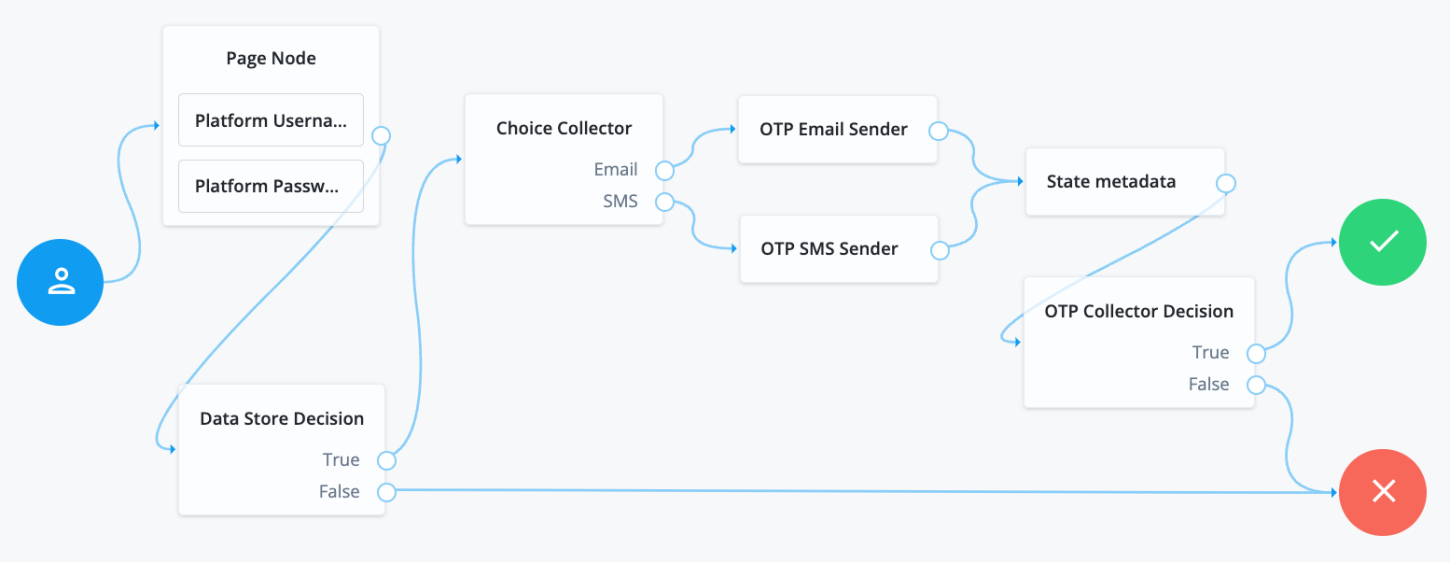
Returns selected attributes from the shared node state as metadata.

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"
          }
        }
      ]
    }
]
```

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.



Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Evaluation continues after the callback.

Properties

Property	Usage
Attributes	Specify one or more shared state attribute names for return.

Success URL node

Sets the redirect URL when authentication succeeds.

> **Note**
>
> Specifying a success URL overrides any `goto` query string parameters.

For more information on how AM determines the redirection URL, and to configure the Validation Service to trust redirection URLs, refer to Configure success and failure redirection URLs.

> **Tip**
>
> The URL is also saved in the `nodeState` object on the `successUrl` key.
> For more information, refer to Customize authentication trees.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

Outcomes

Single outcome path.

Properties

Property	Usage
Start Time Property	Specify the property name containing the time from which to calculate the elapsed time.

Property	Usage
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. For more information, refer to 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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

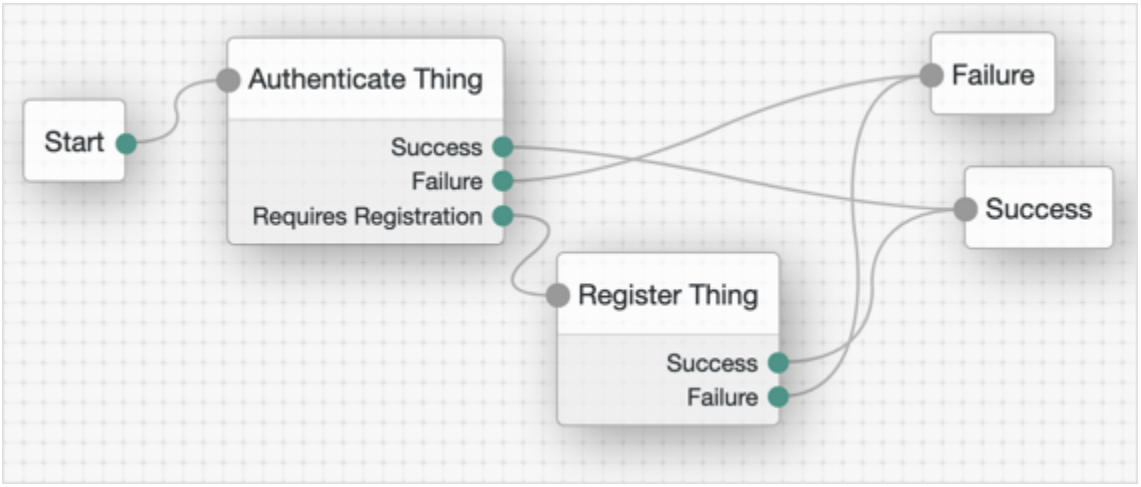
Properties

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✓

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.

Compatibility

Product	Compatible?
PingOne Advanced Identity Cloud	✕
ForgeRock Access Management (self-managed)	✓
Ping Identity Platform (self-managed)	✕

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.