

Authentication node reference

ON THIS PAGE

▸ Basic nodes

Data Store Decision node

Kerberos node

LDAP Decision node

Password Collector node

Username Collector node

Zero Page Login Collector node

▸ Multi-factor nodes

Get Authenticator App node

HOTP Generator node

MFA Registration Options node

OATH Device Storage node

OATH Registration node

OATH Token Verifier node

Opt-out Multi-Factor Authentication node

OTP Collector Decision node

OTP Email Sender node

OTP SMS Sender node

Push Registration node

Push Result Verifier node

Push Sender node

Push Wait node

Recovery Code Collector Decision node

Recovery Code Display node

WebAuthn Authentication node

WebAuthn Device Storage node

WebAuthn Registration node

▸ Risk management nodes

Account Active Decision node

Account Lockout node

Auth Level Decision node

CAPTCHA node

Legacy CAPTCHA node

Modify Auth Level node

▸ Behavioral nodes

Increment Login Count node

Login Count Decision node

▸ Contextual nodes

Certificate Collector node

Certificate User Extractor node

Certificate Validation node

Cookie Presence Decision node

Device Geofencing node

Device Location Match node

Device Match node

Device Profile Collector node

Device Profile Save node

Device Tampering Verification node

Persistent Cookie Decision node

Set Custom Cookie node

Set Persistent Cookie node

▸ Federation nodes

OAuth 2.0 node

OpenID Connect node

Provision Dynamic Account node

Provision IDM Account node

SAML2 Authentication node
Social Facebook node
Social Google node
Social Ignore Profile node
Social Provider Handler node
Write Federation Information node

▸ Identity management nodes

Accept Terms and Conditions node
Attribute Collector node
Attribute Present Decision node
Attribute Value Decision node
Consent Collector node
Create Object node
Create Password node
Display Username node
Identify Existing User node
KBA Decision node
KBA Definition node
KBA Verification node
Passthrough Authentication node
Patch Object node
Platform Password node
Platform Username node
Profile Completeness Decision node
Query Filter Decision node
Required Attributes Present node
Select Identity Provider node
Terms and Conditions Decision node
Time Since Decision node

▸ Utility nodes

Agent Data Store Decision node
Anonymous Session Upgrade node

Anonymous User Mapping node

Choice Collector node

Configuration Provider node

Email Suspend node

Email Template node

Failure URL node

Get Session Data node

Inner Tree Evaluator node

Message node

Meter node

Page node

Polling Wait node

Register Logout Webhook node

Remove Session Properties node

Retry Limit Decision node

Scripted Decision node

Set Session Properties node

State Metadata node

Success URL node

Timer Start node

Timer Stop node

▸ Thing nodes

Authenticate Thing node

Register Thing node

▸ Uncategorized nodes

Debug node

Basic nodes

Data Store Decision node

Verifies that the username and password values match those in the data store configured for the realm.

Outcomes

- True
- False

Properties

This node has no configurable properties.

Alternative nodes

- The LDAP Decision node supports LDAP Behera Password Policies, with separate outcomes for accounts that are locked or password that have expired.

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.

Outcomes

- True
- False

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

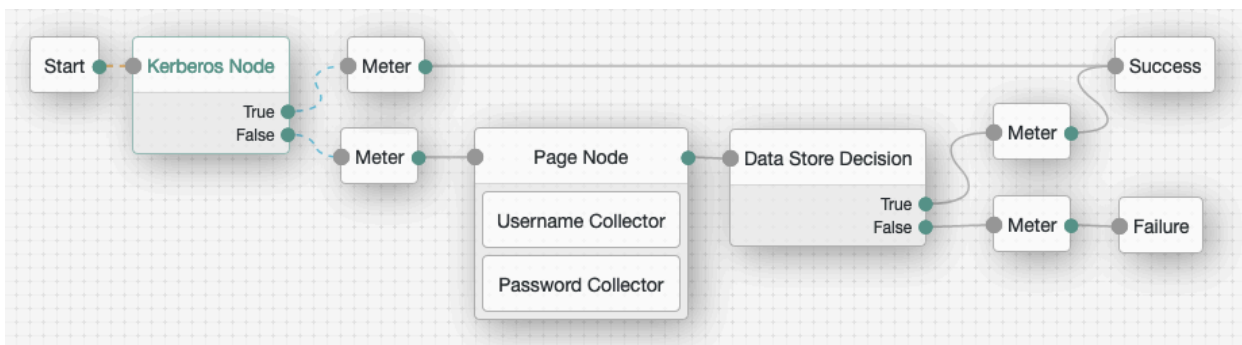
Properties

Property	Usage
Service Principal	<p>Specifies the Kerberos principal for authentication in the format <code>HTTP/AM-DOMAIN@AD-DOMAIN</code>, where <i>AM-DOMAIN</i> corresponds to the host and domain names of the AM instance, and <i>AD-DOMAIN</i> is the domain name of the Kerberos realm (the FQDN of the Active Directory domain). <i>AD-DOMAIN</i> can differ from the domain name for AM.</p> <p>In multi-instance AM deployments, configure <i>AM-DOMAIN</i> as the FQDN or IP address of the load balancer in front of the AM instances.</p> <p>For example, <code>HTTP/AM-LB.example.com@KERBEROSREALM.INTERNAL.COM</code>.</p> <p>For more information, refer to the KB article How do I set up the WDSO authentication module in AM in a load-balanced environment?.</p>
Key Tab File Path	<p>Specifies the full, absolute path of the keytab file for the specified Service Principal.</p> <div data-bbox="608 1115 1401 1563" style="border: 1px solid green; padding: 10px;"> <p>TIP</p> <p>You generate the keytab file using the Windows ktpass utility; for example:</p> <pre style="border: 1px solid gray; padding: 5px;">C:\> ktpass -out fileName.keytab -princ HTTP/openam.example.com@AD_DOMAIN.COM - pass +rdnPass -maxPass 256 -mapuser amKerberos@frdpccloud.com -crypto AES256- SHA1 -ptype KRB5_NT_PRINCIPAL -kvno 0</pre> </div>
Kerberos Realm	<p>Specifies the name of the Kerberos (Active Directory) realm used for authentication.</p> <p>Must be specified in ALL CAPS.</p>
Kerberos Server Name	<p>Specifies the fully qualified domain name, or IP address of the Kerberos (Active Directory) server.</p>

Property	Usage
Trusted Kerberos realms	<p>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.</p> <p>Each trusted Kerberos realm must be specified in all caps.</p>
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	<p>Validates the user against the configured data stores. If the user from the Kerberos token is not found, evaluation continues along the <code>False</code> path.</p> <p>This search uses the <code>Alias Search Attribute Name</code> from the core realm attributes. For more information about this property, refer to User profile.</p>
Is Initiator	<p>When enabled (<code>true</code>), specifies that the node is using <i>initiator</i> credentials, which is the default.</p> <p>When disabled (<code>false</code>), specifies that the node is using <i>acceptor</i> credentials.</p>

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

Verifies that the provided username and password values exist in a specified LDAP user data store, and checks whether they are expired or locked out.

For example, the username and password could be obtained by a combination of the [Username Collector node](#) and [Password Collector node](#), or by using the [Zero Page Login Collector node](#).

Outcomes

True

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

False

The credentials do not 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.

Properties

Property	Usage
Primary LDAP Server (required)	Specify one or more primary directory servers. Specify each directory server in the following format: <i>host:port</i> . For example, <code>directory_services.example.com:389</code> .

Property	Usage
Secondary LDAP Server	<p>Specify one or more secondary directory servers. Specify each directory server in the following format: <i>host:port</i>.</p> <p>Secondary servers are used when none of the primary servers are available.</p> <p>For example, directory_services_backup.example.com:389.</p>
DN to Start User Search <i>(required)</i>	<p>Specify the DN from which to start the user search. More specific DNs, such as <code>ou=sales,dc=example,dc=com</code>, result in better search performance.</p> <p>If multiple entries exist in the store with identical attribute values, ensure this property is specific enough to return only one entry.</p>
Bind User DN, Bind User Password	<p>Specifies the credentials used to bind to the LDAP user data store.</p>
Attribute Used to Retrieve User Profile <i>(required)</i>	<p>Specifies the attribute used to retrieve the profile of a user from the directory server.</p> <p>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.</p>

Property	Usage
Attributes Used to Search for a User to be Authenticated (<i>required</i>)	<p>Specifies the attributes used to match an entry in the directory server to the credentials provided by the user.</p> <p>The default value of <code>uid</code> forms the search filter <code>uid=user</code>. Specifying multiple values such as <code>uid</code> and <code>cn</code> causes the node to forms the complex search filter <code>((uid=user)(cn=user))</code>.</p> <p>Multiple attribute values allow the user to authenticate with any one of the values. For example, if you have both <code>uid</code> and <code>mail</code>, then Barbara Jensen can authenticate with either <code>bjensen</code> or <code>bjensen@example.com</code>.</p> <p>Note that if you have specified multiple attribute values, you must also add those attributes to the <code>Alias Search Attribute Name</code> property when using account lockout. For more information about this property, refer to User profile.</p>
User Search Filter	<p>Specifies an additional filter to append to user searches.</p> <p>For example, searching for <code>mail</code> and specifying a User Search Filter of <code>(objectClass=inetOrgPerson)</code>, causes AM to use <code>(&(mail=address)(objectClass=inetOrgPerson))</code> as the resulting search filter, where <code>address</code> is the mail address provided by the user.</p>
Search Scope	<p>Specifies the extent of searching for users in the directory server.</p> <p>Scope <code>OBJECT</code> means search only the entry specified as the DN to Start User Search, whereas <code>ONELEVEL</code> means search only the entries that are directly children of that object. <code>SUBTREE</code> means search the entry specified and every entry under it.</p> <p>Default: <code>SUBTREE</code></p>

Property	Usage
LDAP Connection Mode	<p>Specifies whether to use SSL or StartTLS to connect to the LDAP user data store. AM must be able to trust the certificates used.</p> <p>Possible values: LDAP , LDAPS , and StartTLS</p> <p>Default: LDAP</p>
Return User DN to DataStore	<p>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</p> <p>uid=demo, ou=people, dc=openam, dc=example, dc=org, AM uses uid=demo to search the data store.</p> <p>Default: Enabled</p>
User Creation Attributes	<p>This list lets you map (external) attribute names from the LDAP directory server to (internal) attribute names used by AM.</p>
Minimum Password Length	<p>Specifies the minimum acceptable password length.</p> <p>Default: 8</p>
LDAP Behera Password Policy Support	<p>When enabled, support interoperability with servers that implement the Internet-Draft, Password Policy for LDAP Directories .</p> <p>Default: Enabled</p>
Trust All Server Certificates	<p>When enabled, blindly trust server certificates, including self-signed test certificates.</p> <p>Default: Disabled</p>

Property	Usage
LDAP Connection Heartbeat Interval	<p>Specifies how often AM should send a heartbeat request to the directory server to ensure that the connection does not remain idle.</p> <p>Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. You can turn this off by setting the value to 0. Set the units for the interval in the LDAP Connection Heartbeat Time Unit property.</p> <p>Note that setting this property to 0 will only ensure default values apply, and will <i>not</i> disable the heartbeat (keepalive) or load balancer availability checks. Disabling these features can only be configured at the global level.</p> <p>Default: 10</p>
LDAP Connection Heartbeat Time Unit	<p>Specifies the time unit for LDAP Connection Heartbeat Interval.</p> <p>Default: seconds</p>
LDAP Operations Timeout	<p>Defines the timeout, in seconds, that AM should wait for a response from the directory server.</p> <p>Default: 0 (means no timeout)</p>
Use mixed case for password change messages	<p>Defines whether password change messages are returned in mixed (sentence) case or transformed to uppercase.</p> <p>By default password reset and password change messages are transformed to upper case. Enable this setting to return messages in sentence case.</p> <p>Default: Disabled</p>

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.

Outcomes

Single outcome path.

Evaluation continues after capturing the password.

Properties

This node has no configurable properties.

Username Collector node

Prompts the user to enter their username.

Outcomes

Single outcome path.

Evaluation continues after capturing the username.

Properties

This node has no configurable properties.

Zero Page Login Collector node

Checks whether selected headers are provided in the incoming authentication request, and if so, uses their values as the provided username and password.

A common use for the Zero Page Login Collector authentication node is to connect the `Has Credentials` outcome connector to the input of a [Data Store Decision node](#), and the `No Credentials` outcome connector to the input of a [Username Collector node](#) followed by a [Password Collector node](#), 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 is transient, persisting only until the next node requiring user interaction.

Outcomes

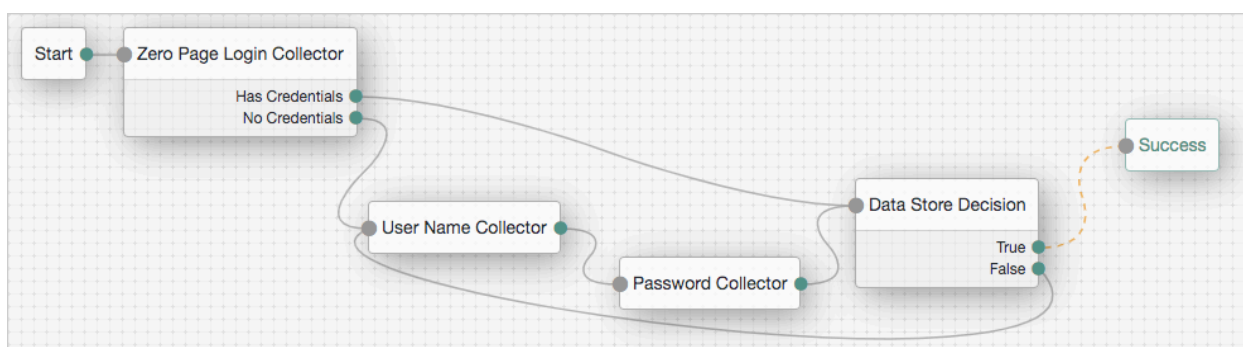
- `Has Credentials`
- `No Credentials`

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.

Properties

Property	Usage
Username Header name	<p>Enter the name of the header that contains the username value.</p> <p>Default: X-OpenAM-Username</p>
Password Header name	<p>Enter the name of the header that contains the password value.</p> <p>Default: X-OpenAM-Password</p>
Allow without referer	<p>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.</p> <p>If disabled, a Referer HTTP header must be present in the incoming request, and the value must appear in the Referer allowlist property.</p> <p>Default: Enabled</p>
Referer Whitelist	<p>Specify a list of URLs allowed in the Referer HTTP header of incoming requests. Incoming requests containing a Referer HTTP header value not specified in the allowlist causes evaluation to continue along the No Credentials outcome path.</p> <div style="border: 1px solid #0070C0; padding: 5px; margin-top: 10px;"> <p>NOTE</p> <p>You must disable the Allow Without Referer property for the referer allowlist property to take effect.</p> </div>

Example



Multi-factor nodes

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>
```

Outcomes

Single outcome path.

Properties

Property	Usage
Get App Authenticator Message	Localized title for the node. The key is the language, such as <code>en</code> or <code>fr</code> , and the value is the message to display. Default: Get the app from the <code>{{appleLink}}</code> or on <code>{{googleLink}}</code> .
Continue Label	Localized text to use on the Continue button. The key is the language, such as <code>en</code> or <code>fr</code> , and the value is the message to display.

Property	Usage
Apple App Store URL	<p>Specifies the URL to download your authenticator application from the Apple App Store. The default value points to the ForgeRock Authenticator application for iOS.</p> <p>Default: https://itunes.apple.com/app/forgerock-authenticator/id1038442926</p>
Google Play URL	<p>Specifies the URL to download your authenticator application from the Google Play Store. The default value points to the ForgeRock Authenticator application for Android.</p> <p>Default: https://play.google.com/store/apps/details?id=com.forgerock.authenticator</p>

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](#)

Outcomes

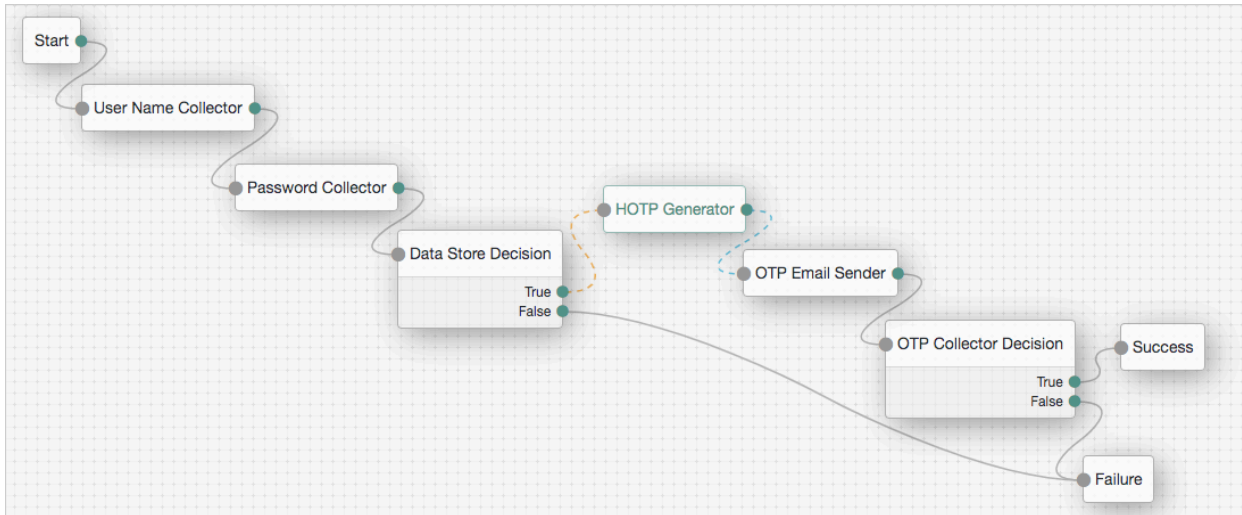
Single outcome path.

Properties

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

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](#) and a [Push Sender node](#) earlier in the flow to obtain these.

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.

Property	Usage
Message	<p>Localized text to use as the title of the screen.</p> <p>The key is the language, such as <code>en</code> or <code>fr</code>, and the value is the message to display.</p>
Register Device	<p>Localized text to use on the Register Device button.</p> <p>The key is the language, such as <code>en</code> or <code>fr</code>, and the value is the message to display.</p>
Get Authenticator App	<p>Localized text to use on the Get Authenticator App button.</p> <p>The key is the language, such as <code>en</code> or <code>fr</code>, and the value is the message to display.</p>
Skip this Step	<p>Localized text to use on the Skip this Step button.</p> <p>The button and the outcome only appear if the Remove 'skip' option is not enabled.</p> <p>The key is the language, such as <code>en</code> or <code>fr</code>, and the value is the message to display.</p>
Opt-out	<p>Localized text to use on the Opt-Out button.</p> <p>The button and the outcome only appear if the Remove 'skip' option is not enabled.</p> <p>Note that this node does not change the user's profile. Connect the Opt-out outcome to an <u>Opt-out Multi-Factor Authentication node</u> to persist the option in the user's profile.</p> <p>The key is the language, such as <code>en</code> or <code>fr</code>, and the value is the message to display.</p>

Example

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



On this page you can choose to register, skip or opt-out the second factor authentication method selected to protect your account. If you "Skip", an MFA method will not be registered now, but you will be prompted again on your next login. Otherwise, if you "Opt out", an MFA method will not be registered now and you will not be asked again. This choice is not recommended.

Register Device

Get the App

Skip this step

Opt-out

OATH Device Storage node

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

Compatibility

Product	Compatible?
ForgeRock Identity Cloud	Yes
ForgeRock Access Management (self-managed)	Yes
ForgeRock Identity Platform (self-managed)	Yes

Authenticators

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

- The [ForgeRock Authenticator](#) app for Android and iOS.
- Third-party authenticator apps that support the following open standards:

- [RFC 4226](#): HMAC-Based One-Time Password (HOTP)
- [RFC 6238](#): Time-Based One-Time Password (TOTP)

Inputs

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

Dependencies

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

Configuration

This node has no configurable properties.

Outputs

This node doesn't change the shared state.

Outcomes

Success

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

Failure

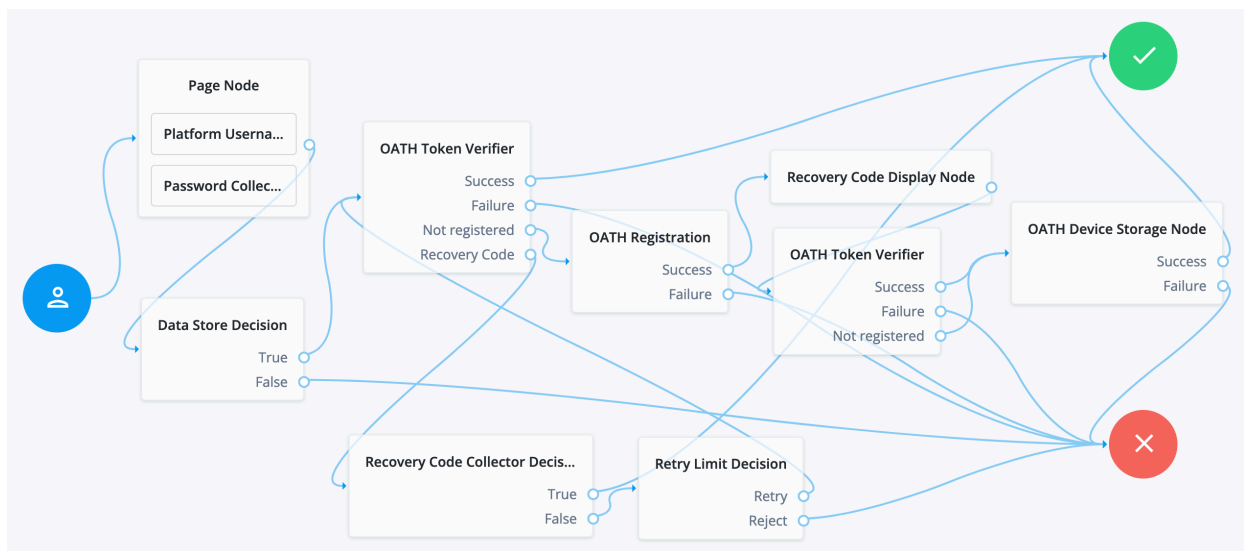
Any other case.

Errors

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

Example

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



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

OATH Registration node

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.

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
- Password Collector node

- [Data Store Decision node](#)

Connect the OATH Registration node's `Success` outcome path to the [OATH Token Verifier node](#) to continue with OTP verification.

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.

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

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.

Property	Usage
Logo Image URL	<p>The location of an image to download and display as the issuer's logo within the authenticator application.</p> <div data-bbox="608 315 1399 613" style="border: 1px solid #00aaff; padding: 10px;"> <p>NOTE</p> <p>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.</p> </div>
Generate Recovery Codes	<p>If enabled, recovery codes are generated and stored in the successful outcome's transient state.</p> <p>Use the Recovery Code Display node to display the codes to the user for safekeeping.</p>
One Time Password Length	<p>The length of the generated OTP in digits.</p> <p>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.</p>
Minimum Secret Key Length	<p>Number of hexadecimal characters allowed for the Secret Key.</p>
OATH Algorithm	<p>Specify the algorithm your device uses to generate the OTP:</p> <p>HOTP</p> <p>HOTP uses a counter value that is incremented every time a new OTP is generated.</p> <p>TOTP (default)</p> <p>TOTP generates a new OTP every few seconds as specified by the TOTP Time Step Interval value.</p> <p>If this is set to HOTP, set the same value in the OATH Token Verifier node.</p>

Property	Usage
TOTP Time Step Interval	<p>The length of time that an OTP is valid in seconds.</p> <p>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.</p> <p>The default value is 30.</p>
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	<p>The message with instructions to scan the QR code to register the device.</p> <p>Click Add. Enter the message locale in the <code>Key</code> field; for example, <code>en-gb</code>. Enter the message to display to the user in the <code>Value</code> field.</p>
Store device data in shared state	If enabled, the device is not stored directly in the user profile upon successful completion of the node. Instead, the device information is added into the shared node state on the <code>oathDeviceData</code> key. Use the OATH Device Storage node to store the device information in the user profile.

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.

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.

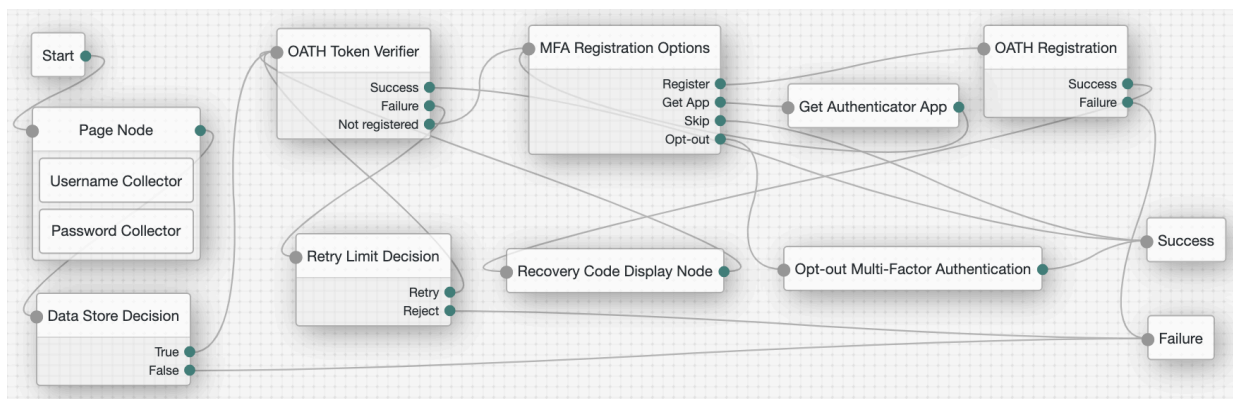
Properties

Property	Usage
OATH Algorithm	<p>Specify the algorithm your device uses to generate the OTP:</p> <p>HOTP HOTP uses a counter value that is incremented every time a new OTP is generated.</p> <p>TOTP (default) TOTP generates a new OTP every few seconds as specified by the TOTP Time Step Interval value.</p> <p>If this is set to HOTP, you need to set the same value in the OATH Registration node.</p>

Property	Usage
HOTP Window Size	<p>This property sets the window that the OTP device and the server counter can be out of sync.</p> <p>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.</p> <p>The default value is 100.</p>
TOTP Time Step Interval	<p>The length of time that an OTP is valid, in seconds.</p> <p>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.</p> <p>The default value is 30.</p>
TOTP Time Steps	<p>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.</p> <p>For example, with a time step of 1, the server permits either the previous, the current, or the next code.</p> <p>The default value is 2.</p>
TOTP Hash Algorithm	<p>The HMAC hash algorithm to be used to generate the OTP codes. ForgeRock Authenticator (OATH) supports SHA1, SHA256, and SHA512.</p>
TOTP Maximum Allowed Clock Drift	<p>Number of time steps a client can be out of sync with the server before manual resynchronization is required.</p> <p>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.</p> <p>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.</p> <p>The default value is 5.</p>

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](#) and a [Push Sender node](#) earlier in the flow to obtain these.

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.

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 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`

▼ [How Do I Configure Advanced Server Properties?](#)

- 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](#).

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.
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.

Property	Usage
Mail Server Secure Connection	<p>Specifies how to connect to the mail server.</p> <p>If a secure method is specified, AM must trust the server certificate of the mail server.</p> <p>The possible values for this property are:</p> <ul style="list-style-type: none"> • NON SSL/TLS • SSL/TLS • Start TLS <p>Default: SSL/TLS</p>
Gateway Implementation Class	<p>Specifies the class the node uses to send SMS and email messages. A custom class must implement the <code>com.sun.identity.authentication.modules.hotp.SMSGateway</code> interface.</p> <p>Default: <code>com.sun.identity.authentication.modules.hotp.DefaultSMSGatewayImpl</code></p>

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` .

Outcomes

Single outcome path.

Properties

Property	Usage
Mail Server Host Name <i>(required)</i>	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 <i>(required)</i>	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: <code>telephoneNumber</code>
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 <code>en-uk</code> , 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 <code>en-uk</code> , 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	<p>Specifies how to connect to the mail server.</p> <p>If a secure method is specified, AM must trust the server certificate of the mail server.</p> <p>The possible values for this property are:</p> <ul style="list-style-type: none"> • NON SSL/TLS • SSL/TLS • Start TLS <p>Default: SSL/TLS</p>
Gateway Implementation Class	<p>Specifies the class the node uses to send SMS and email messages. A custom class must implement the <code>com.sun.identity.authentication.modules.hotp.SMSGateway</code> interface.</p> <p>Default: <code>com.sun.identity.authentication.modules.hotp.DefaultSMSGatewayImpl</code></p>

Push Registration node

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

For more information, refer to [MFA: Push authentication](#).

The node requires the username of the identity to update; for example, by using a [Username Collector node](#).

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](#).

Outcomes

- Success

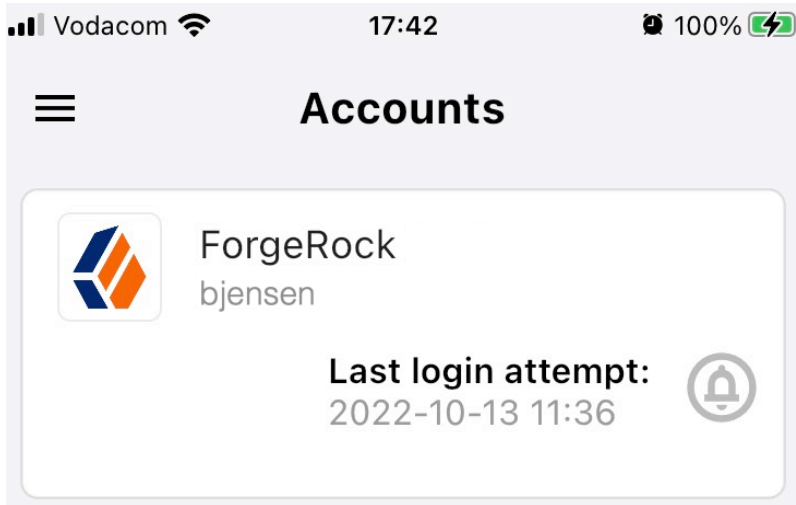
- Failure
- Time Out

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	<p>Specify an identifier so that the user knows which service their account relates to.</p> <p>The value is displayed by the authenticator application:</p>  <p>For example, Example Inc. or the name of your application.</p>
Account Name	<p>Specifies the profile attribute to display as the username in the authenticator application.</p> <p>If not specified, or if the specified profile attribute is empty, their username is used.</p>
Registration Response Timeout	<p>Specify the number of seconds to wait for a response from the authenticator.</p> <p>If the specified time is reached, evaluation continues along the Time Out outcome path.</p>

Property	Usage
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	<p>Specify whether push-specific recovery codes should be generated. If enabled, recovery codes are generated and stored in transient state if registration was successful.</p> <p>Use the Recovery Code Display node to display the codes to the user for safe keeping.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>IMPORTANT</p> <p>Generating recovery codes overwrites all existing push-specific recovery codes.</p> <p>Only the most recent set of recovery codes can be used for authentication if a device has been lost or stolen.</p> </div>
QR code message	<p>The message with instructions to scan the QR code to register the device.</p> <p>Click Add. Enter the message locale in the Key field; for example, <code>en-gb</code>. Enter the message to display to the user in the Value field.</p>

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.

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](#).

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](#).

Outcomes

- Sent
- Not Registered
- Skipped

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

If the user does not 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.

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.

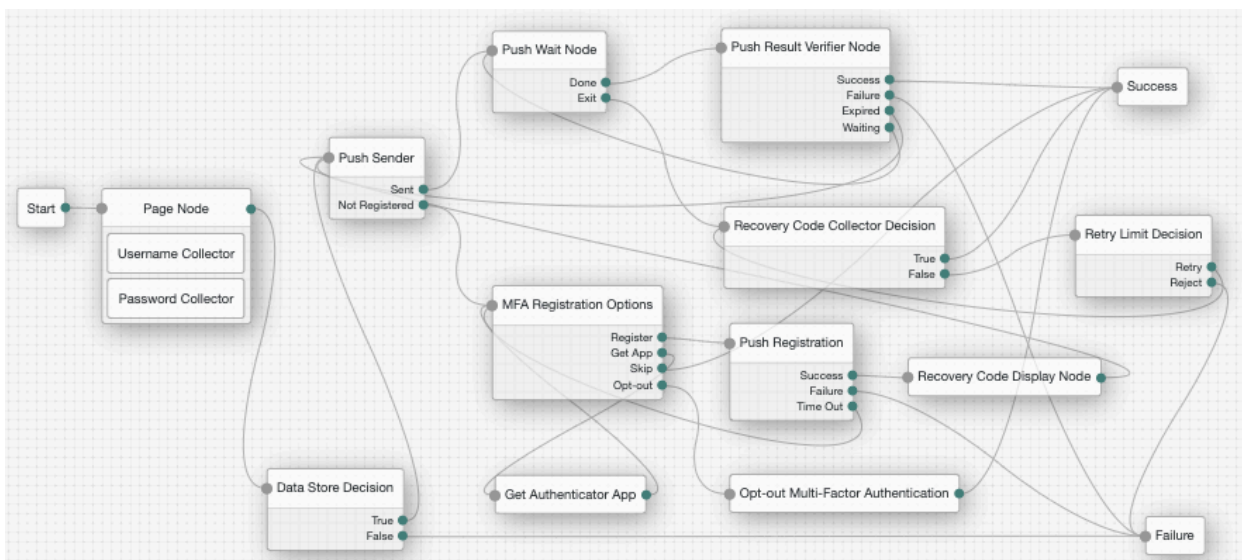
Property	Usage
User Message	<p>Specifies the optional message to send to the user.</p> <p>You can provide the message in multiple languages by specifying the locale in the <code>KEY</code> field; for example, <code>en-US</code>.</p> <p>The locale selected for display is based on the user's locale settings in their browser.</p> <p>Messages provided in the node override the defaults provided by AM. For information about customizing and translating the default messages, refer to Internationalization.</p> <p>The following variables can be used in the <code>VALUE</code> field:</p> <p><code>{{user}}</code> Replaced with the username value of the account registered in the ForgeRock Authenticator application, for example <i>Demo</i>.</p> <p><code>{{issuer}}</code> Replaced with the issuer value of the account registered in the ForgeRock Authenticator application, for example <i>ForgeRock</i>.</p> <p>Example: Login attempt from <code>{{user}}</code> at <code>{{issuer}}</code>.</p>
Remove 'skip' option	<p>Enable this option in the node to make the push authentication mandatory.</p> <p>When disabled, the user can skip the push authentication requested by the node, and evaluation continues along the <code>Skipped</code> outcome path.</p> <p>Default: Disabled</p> <div data-bbox="608 1749 1399 2056" style="border: 1px solid #0070C0; padding: 10px;"> <p>NOTE</p> <p>Nodes in authentication trees are not affected by the Two Factor Authentication Mandatory property, available at Realms > <i>Realm Name</i> > Authentication > Settings > General, as it only applies to modules within authentication chains.</p> </div>

Property	Usage
Share Context info	<p>If enabled, context data such as <code>remoteIp</code> , <code>userAgent</code> , and <code>location</code> are included in the notification payload.</p> <p>For example:</p> <pre data-bbox="608 439 1399 952"> { "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" } </pre> <p>For the location attribute to be set, the flow must contain a Device Profile Collector node with Collect Device Location enabled.</p>
Custom Payload Attributes	<p>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 <code>NodeProcessException</code> is thrown.</p> <p>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.</p>

Property	Usage
Push Type	<p>Select the type of the push notification that must be processed before the notification is sent.</p> <p>Possible values are:</p> <p>Tap to Accept (default) Requires the user to tap to accept.</p> <p>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.</p> <p>Use Biometrics to Accept Requires the user's biometric authentication to process the notification.</p>

Example

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



▼ [Node connections](#)

List of node connections

Source node	Outcome path	Target node
-------------	--------------	-------------

Source node	Outcome path	Target node
Page Node containing: Username Collector, Password Collector	→	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.

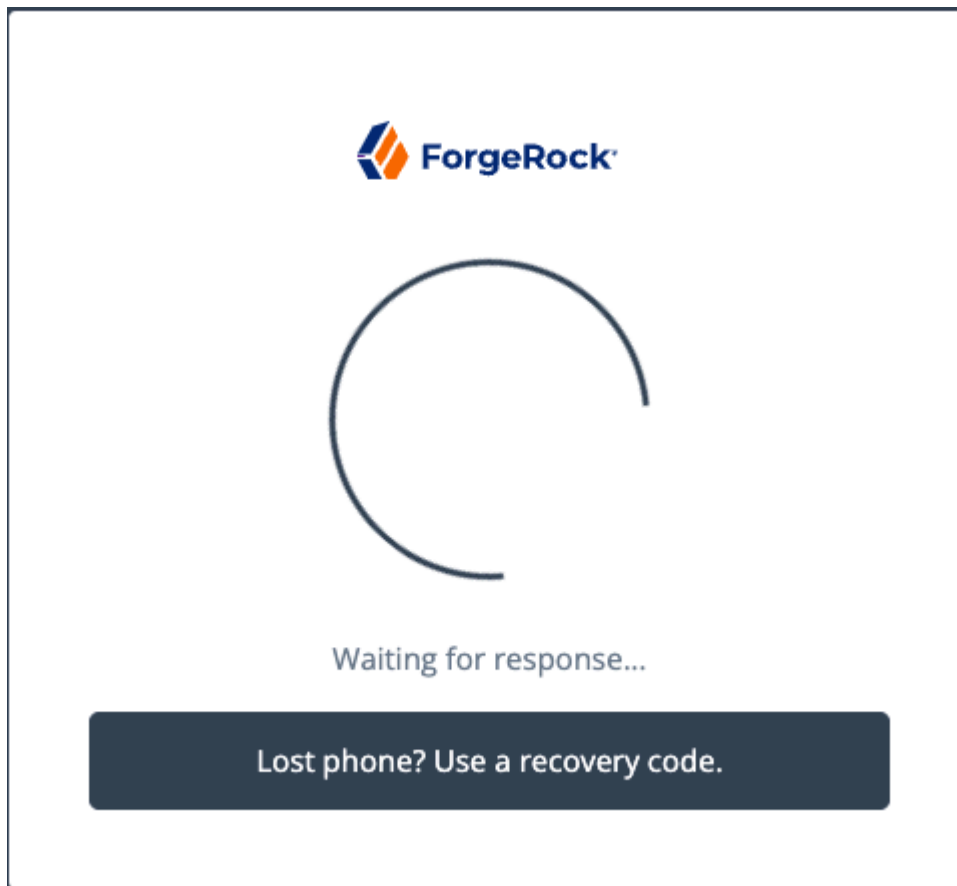
TIP

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](#).

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](#).

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 <code>{{time}}</code> 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 <code>{{challenge}}</code> 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.
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](#).

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](#).

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.

Outcomes

Single outcome path.

Properties

This node has no configurable properties.

Example

The following shows example output of this node:



Device sign-in is enabled

Now you can use your device as a second step when signing in to your account.

Don't get locked out of your account

If you lose your device, or don't have it with you, a recovery code is the only way to sign in to your account with 2-step verification enabled. It's strongly recommended that you print and store these codes in a safe place. **Each code can only be used once.**

- | | |
|---------------|----------------|
| 1. SPPiJU48lQ | 6. fX0A5Xodem |
| 2. bpuF60ZnPc | 7. jsb8eGDUFy |
| 3. 9Dklyyu92K | 8. aSWNa1VKth |
| 4. SMCcghBlfU | 9. glkCeZ32Jn |
| 5. CLlrJdjggW | 10. QAw0U6uW90 |

Copy

Print

Done

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 shared node state must a username. You can use a [Username Collector node](#) for this.

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 user 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.

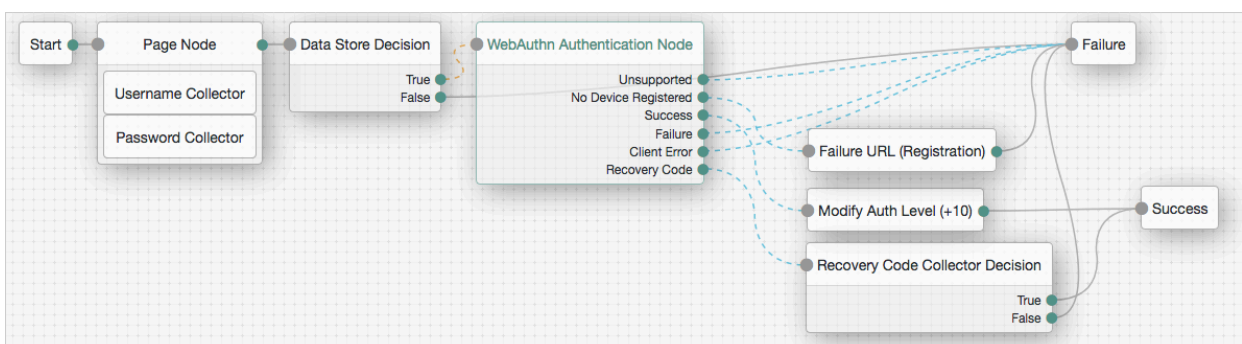
Properties

Property	Usage
Relying party identifier	<p>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, <code>am.example.com</code>.</p> <p>Specify an alternative domain if your AM instances are behind a load balancer, for example.</p>
Origin domains	<p>Specifies a list of fully qualified URLs to accept as the origin of incoming requests.</p> <p>If left empty, AM accepts any incoming domain.</p>
User verification requirement	<p>Specifies the required level of user verification .</p> <p>The available options are:</p> <p>REQUIRED</p> <p>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.</p> <p>PREFERRED</p> <p>Use of an authenticator that verifies the identity of the user is preferred, but if none are available any authenticator is accepted.</p> <p>DISCOURAGED</p> <p>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.</p>
Allow recovery codes	<p>Specify whether to allow the user to enter one of their recovery codes instead of performing an authentication gesture.</p> <p>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.</p>

Property	Usage
Timeout	<p>Specify the number of seconds to wait for a response from an authenticator.</p> <p>If the specified time is reached, evaluation continues along the <code>Client error</code> outcome path, and a relevant message is stored in the <code>WebAuthenticationDOMException</code> property of the shared state.</p>
Username from device	<p>Specifies whether AM requests that the device provides the username.</p> <p>When enabled, if the device is unable to store or provide usernames, the node fails and evaluation continues along the <code>Failure</code> path.</p> <p>For information on using this property for usernameless authentication with ForgeRock Go, refer to Configure usernameless authentication with ForgeRock Go.</p>
Return challenge as JavaScript	<p>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.</p> <p>If disabled, the node returns the challenge and associated data in a metadata callback. A custom UI, for example an application using the ForgeRock SDKs, uses the information from the callback to interact with the WebAuthn API on AM's behalf.</p>

Example

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

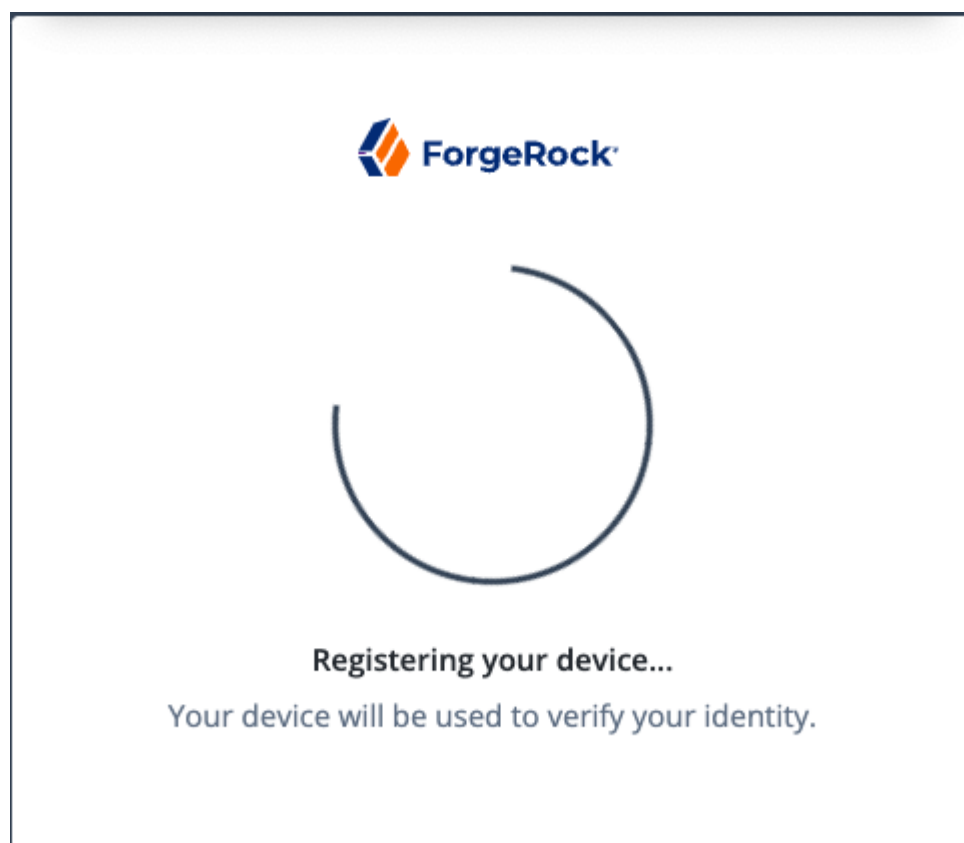


After verifying the users credentials against the configured data store, evaluation continues to the WebAuthn Authentication node.

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](#)^[7], 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.

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	<p>Specify whether WebAuthn device recovery codes should be generated.</p> <p>If enabled, recovery codes are generated and stored in the transient node state, and stored alongside the device profile.</p> <p>Use the Recovery Code Display node to display the codes to the user for safe keeping.</p> <div style="border: 1px solid purple; padding: 5px; margin-top: 10px;"> <p>IMPORTANT</p> <p>Generating recovery codes overwrites all existing WebAuthn device recovery codes for the device.</p> <p>Only the most recent set of recovery codes can be used for authentication if a device has been lost or stolen.</p> </div>
Maximum Saved Devices	<p>Specify the maximum number of WebAuthn devices to save in a user's profile.</p> <p>Set this property to 0 if you do not want to limit the number of devices.</p> <p>When this property is greater than zero, the <code>Exceed Device Limit</code> outcome path becomes available.</p>

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.

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, <code>Example Inc.</code>
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 <code>am.example.com</code> . Specify an alternative domain if your AM instances are behind a load balancer, for example.

Property	Usage
Origin domains	<p>Specifies a list of fully qualified URLs to accept as the origin of incoming requests.</p> <p>If left empty, AM accepts any incoming domain.</p>
User verification requirement	<p>Specifies the required level of user verification .</p> <p>The available options are:</p> <p>REQUIRED</p> <p>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.</p> <p>PREFERRED</p> <p>Use of an authenticator that verifies the identity of the user is preferred, but if none are available any authenticator is accepted.</p> <p>DISCOURAGED</p> <p>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.</p>

Property	Usage
Preferred mode of attestation	<p>Specifies whether AM requires that the authenticator provides attestation statements.</p> <p>The available options are:</p> <p><i>NONE</i></p> <p>AM does not require the authenticator to provide attestation statements. If the authenticator does send attestation statements, AM <i>will not</i> verify them, and will not fail the process.</p> <p><i>INDIRECT</i></p> <p>AM does not require the authenticator to provide attestation statements. If the authenticator does send attestation statements, AM <i>will</i> verify them, and will fail the process if they fail verification.</p> <p><i>DIRECT</i></p> <p>AM requires the authenticator provides attestation statements, and <i>will</i> verify them. The process will fail if the attestation statements cannot be verified.</p> <p>AM supports the following attestation formats:</p> <ul style="list-style-type: none"> • None  • Android SafetyNet  • Packed  • FIDO U2F  • TPM  <div style="border: 1px solid purple; padding: 10px; margin-top: 10px;"> <p>IMPORTANT</p> <p>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.</p> <p>Specifically, AM <i>does not</i> currently support:</p> <ul style="list-style-type: none"> • android-safetynet  • android-key  </div>
Accepted signing algorithms	Specify the algorithms authenticators can use to sign their assertions.

Property	Usage
Authentication attachment	<p>Specifies whether AM requires that the authenticator is a particular attachment type.</p> <p>There are two types of authenticator attachments:</p> <ul style="list-style-type: none"> • An authenticator that is built-in to the client device is labeled a <i>platform attachment</i>. <p>A fingerprint scanner built-in to a phone or laptop is an example of a platform attachment authenticator.</p> <ul style="list-style-type: none"> • An authenticator that can roam, or move, between different client devices is labeled a <i>cross-platform attachment</i>. <p>A USB hardware security key is an example of a cross-platform attachment authenticator.</p> <p>The available options are:</p> <p><i>UNSPECIFIED</i> AM accepts any attachment type.</p> <p><i>PLATFORM</i> The authenticator must be a <i>platform</i> attachment type. The client should not activate other authenticator types for registration.</p> <p><i>CROSS_PLATFORM</i> The authenticator must be a <i>cross-platform</i> attachment type. The client should not activate other authenticator types for registration.</p>

Property	Usage
Trust Store alias	<p>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.</p> <p>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.</p> <p>The value is also appended to the string <code>am.authentication.nodes.webauthn.truststore.</code> to form the dynamic secret ID used to map the certificate chains.</p>
Enforce revocation check	<p>Specifies whether to enforce certificate revocation checks. When enabled, then any attestation certificate's trust chain <i>MUST</i> have a CRL or OCSP entry that can be verified by AM during processing.</p> <p>When disabled, certificates are not checked for revocation. You must ensure expired or revoked certificates are manually removed.</p>
Timeout	<p>Specify the number of seconds to wait for a response from an authenticator.</p> <p>If the specified time is reached, evaluation continues along the <code>Client error</code> outcome path, and a relevant message is stored in the <code>WebAuthenticationDOMException</code> property of the shared state.</p>
Limit registrations	<p>Specify whether the same authenticator can be registered multiple times.</p> <p>If enabled, the client should not activate an authenticator that is already registered for registration.</p>

Property	Usage
Generate recovery codes	<p>Specify whether WebAuthn-specific recovery codes should be generated. If enabled, recovery codes are generated and stored in transient state if registration was successful.</p> <p>Use the Recovery Code Display node to display the codes to the user for safe-keeping.</p> <p>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.</p> <div data-bbox="608 792 1401 1115" style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>IMPORTANT</p> <p>Generating recovery codes will overwrite all existing WebAuthn-specific recovery codes.</p> <p>Only the most recent set of recovery codes can be used for authentication if a device has been lost or stolen.</p> </div>
Store data in transient state	<p>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 <code>webauthnData</code> .</p> <p>In addition to the information provided by the device, the type of attestation achieved; for example, <code>BASIC</code> , <code>CA</code> , <code>SELF</code> and so on, is stored in the transient node state, using the key <code>webauthnAttestationType</code> .</p> <div data-bbox="608 1581 1401 1731" style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>WARNING</p> <p>The amount of data involved can be large. Only enable this option if you intend to analyze it.</p> </div>

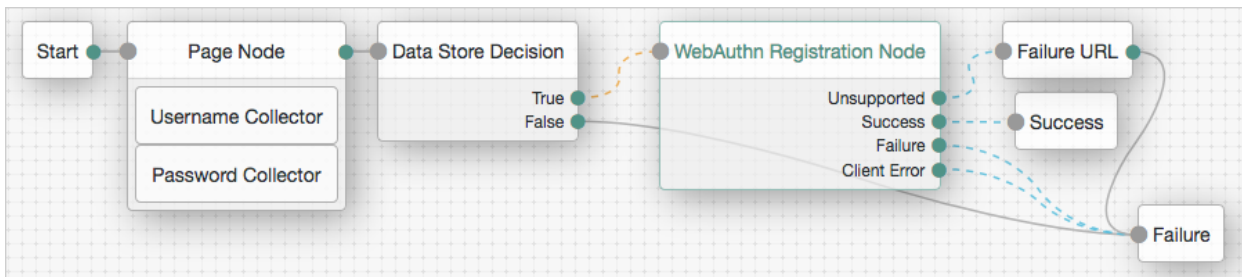
Property	Usage
Store device data in transient state	<p>Specify whether the information about the device required for WebAuthn is stored in the transient node state rather than saved immediately to the user's profile.</p> <p>Enable this option if you intend to make decisions in scripts, and have enabled 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.</p> <div data-bbox="608 658 1401 949" style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p>IMPORTANT</p> <p>Do not alter the data while it is in the transient node state, nor when saved to a user's profile.</p> <p>Modifying the device data will likely cause the device to be unable to authenticate.</p> </div> <p>Use the WebAuthn Device Storage node to write the device data to the user's profile when this option is enabled.</p> <p>When disabled, device data is written automatically to the user's profile when registration is successful.</p>
Username to device	<p>Specifies whether AM requests that the device stores the user's username.</p> <p>When enabled, if the device is unable to store or provide usernames, the node will fail and results in the <i>Failure</i> outcome.</p> <p>For information on using this property for usernameless authentication with ForgeRock Go, refer to Configure usernameless authentication with ForgeRock Go.</p>

Property	Usage
Shared state attribute for display name	<p>Specifies a variable in shared node state that contains a display name for the user; for example, their full name, or email address.</p> <p>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.</p> <p>If not specified, or the variable is not found in shared state, the username is used.</p> <p>For information on using this property for usernameless authentication with ForgeRock Go, refer to Configure usernameless authentication with ForgeRock Go.</p>
Return challenge as JavaScript	<p>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.</p> <p>If disabled, the node returns the challenge and associated data in a metadata callback. A custom UI, for example, an application using the ForgeRock SDKs, uses the information from the callback to interact with the WebAuthn API on AM's behalf.</p>

Property	Usage
<p>Maximum Saved Devices</p>	<p>Specifies the maximum number of WebAuthn devices stored in the user's profile.</p> <p>Set this property to 0 if you do not want to limit the number of devices.</p> <p>When this property is greater than zero, the <code>Exceed Device Limit</code> outcome path becomes available.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>IMPORTANT</p> <p>You can only limit the number of devices stored in the user's profile.</p> <p>If the Store device data in transient state property is enabled then the node is unable to limit the number of devices, and the <code>Exceed Device Limit</code> outcome path is not displayed.</p> <p>In this case, specify the maximum number of saved devices in the WebAuthn Device Storage node.</p> </div>

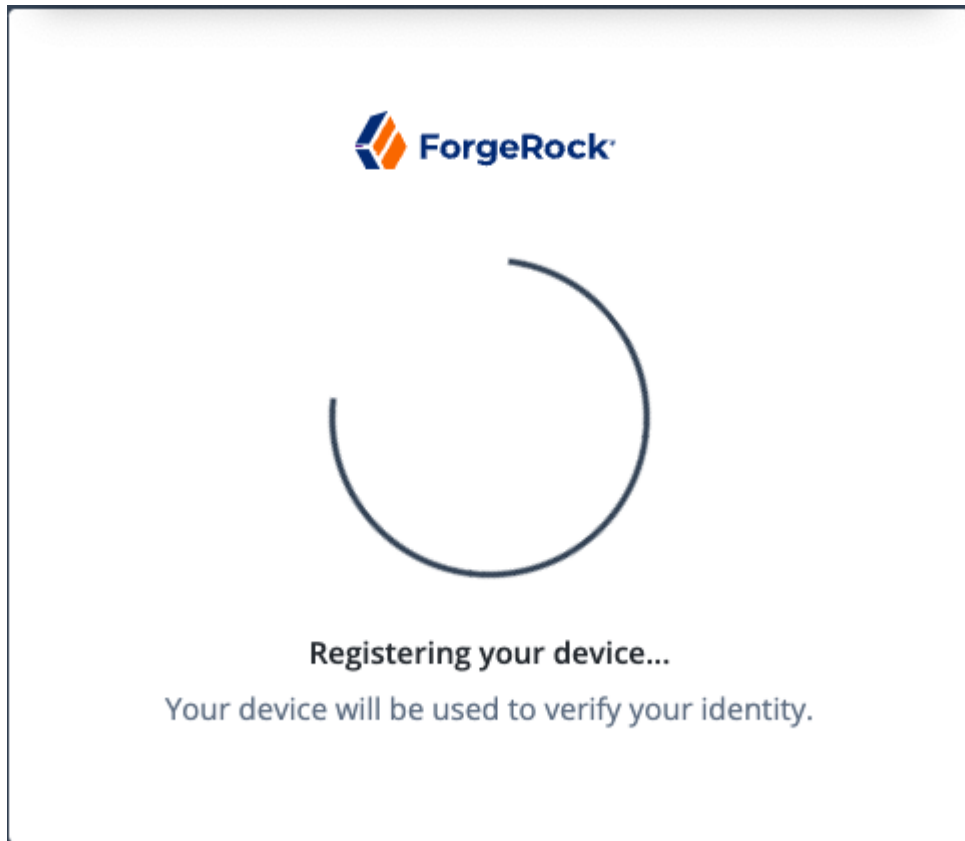
Example

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](#)^[2], 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.

For more information, refer to [Account lockout for trees](#).

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 trees](#).

Outcomes

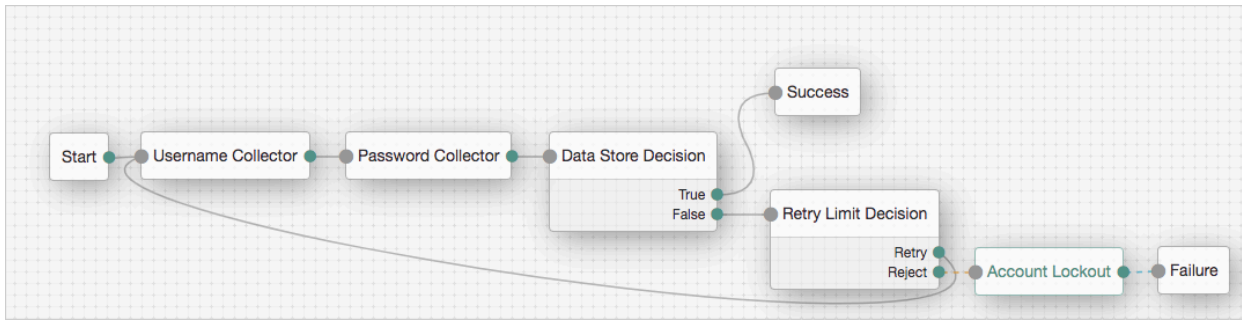
Single outcome path.

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.

Outcomes

- True
- False

Properties

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

Outcomes

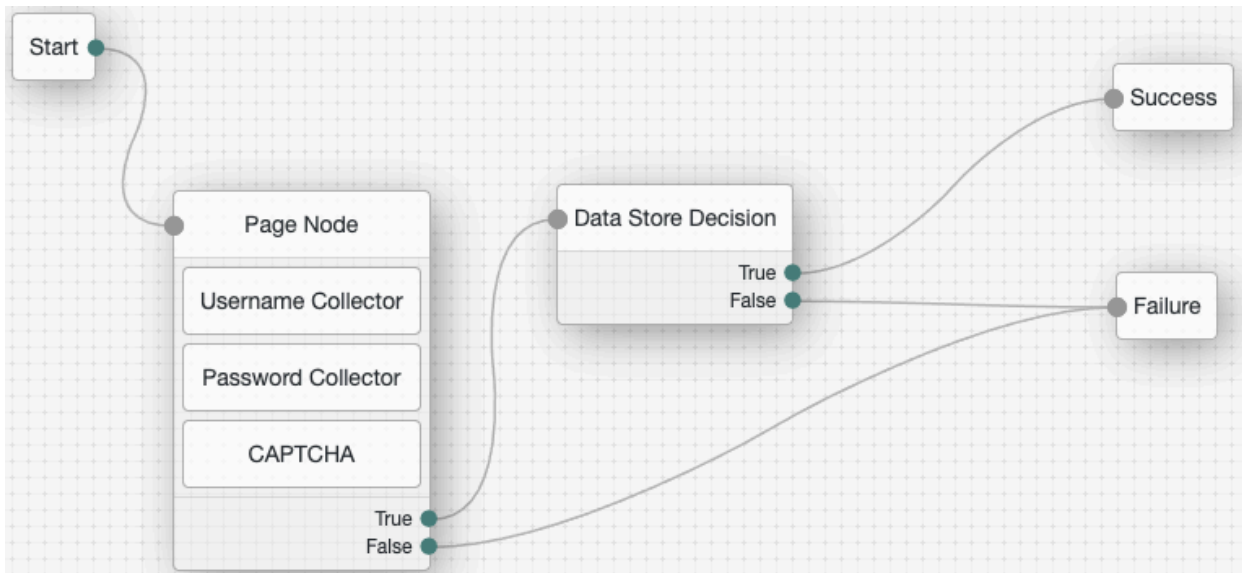
- True (*success*)
- False (*failure*)

Properties

Property	Usage
CAPTCHA Site Key <i>(required)</i>	The CAPTCHA site key supplied by the CAPTCHA provider when you sign up for access to the API.
CAPTCHA Secret Key <i>(required)</i>	The CAPTCHA secret key supplied by the CAPTCHA provider when you sign up for access to the API.
CAPTCHA Verification URL <i>(required)</i>	<p>The URL used to verify the CAPTCHA submission.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> • Google: <code>https://www.google.com/recaptcha/api/siteverify</code> • hCaptcha: <code>https://hcaptcha.com/siteverify</code>
CAPTCHA API URL <i>(required)</i>	<p>The URL of the JavaScript that loads the CAPTCHA widget.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> • Google: <code>https://www.google.com/recaptcha/api.js</code> • hCaptcha: <code>https://hcaptcha.com/1/api.js</code>
Class of CAPTCHA HTML Element	<p>The class of the HTML element required by the CAPTCHA widget.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> • Google: <code>g-recaptcha</code> • hCaptcha: <code>h-captcha</code>
ReCaptcha V3 node	If you're using Google reCAPTCHA, specify whether it's v2 or v3. Turn on for v3.

Property	Usage
Score Threshold	<p>If you're using Google reCAPTCHA v3, or hCaptcha, enter a score threshold.</p> <p>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.</p> <p>A score of 1.0 is likely a good user interaction, while 0.0 is likely to be a bot.</p> <p>The threshold you set here determines whether to allow or deny access, based on the score returned by the CAPTCHA provider.</p> <p>Start with a threshold of 0.5.</p> <p>For more information about score thresholds, refer to the Google documentation.</p>

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.

This node has been superseded by the [CAPTCHA node](#). Use that node instead.

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: <ul style="list-style-type: none"> • Google: <code>https://www.google.com/recaptcha/api/siteverify</code> • hCaptcha: <code>https://hcaptcha.com/siteverify</code>
CAPTCHA API URL (required)	The URL of the JavaScript that loads the CAPTCHA widget. Possible values are: <ul style="list-style-type: none"> • Google: <code>https://www.google.com/recaptcha/api.js</code> • hCaptcha: <code>https://hcaptcha.com/1/api.js</code>
Class of CAPTCHA HTML Element	The class of the HTML element required by the CAPTCHA widget. Possible values are: <ul style="list-style-type: none"> • Google: <code>g-recaptcha</code> • hCaptcha: <code>h-captcha</code>

Modify Auth Level node

Increases or decreases the current authentication level value.

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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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

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.

NOTE

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

Add the [Increment Login Count node](#) to your login flows, so this node has the data to trigger a decision.

Properties

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

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](#).

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	<p>Specifies how to collect the certificate from the request. Possible values are:</p> <p>Request Look for the certificate in the request. Use this value if TLS termination happens at the container where AM runs.</p> <p>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.</p> <p>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.</p> <p>Default: <code>Either</code></p>
HTTP Header Name for the Client Certificate	<p>Specifies the name of the HTTP header containing the certificate when the Certificate Collection Method property is configured to <code>Header</code> or <code>Either</code>.</p> <p>Default: No value specified.</p>
Trusted Remote Hosts	<p>Specifies a list of IP addresses trusted to supply certificates on behalf of the authenticating client, such as load balancers doing TLS termination.</p> <p>If no value is specified, AM rejects certificates supplied by remote hosts. If you specify the <code>any</code> value, AM trusts certificates on behalf of the authenticating client supplied by any remote host.</p> <p>Default: No value specified.</p>

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.

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	<p>Specifies the field in the certificate that AM uses to search for the user in the identity store. Possible values are:</p> <ul style="list-style-type: none">• Subject DN• Subject CN• Subject UID• Email Address• Other• None <p>If you select <code>Other</code>, provide an attribute name in the Other Certificate Field Used to Access User Profile property.</p> <p>Select <code>None</code> if you want to specify an alternate way of looking up the user profile in the SubjectAltNameExt Value Type to Access User Profile property.</p> <p>Default: Subject CN</p>
Other Certificate Field Used to Access User Profile	<p>Specifies a custom certificate field to use as the base of the user search.</p>

Property	Usage
SubjectAltNameExt Value Type to Access User Profile	<p>Specifies how to look up the user profile:</p> <p>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.</p> <p>RFC822Name AM looks up the user profile using the value of the RFC822Name field.</p> <p>UPN AM looks up the user profile as the User Principal Name attribute used in Active Directory.</p> <p>Default: None</p>

Certificate Validation node

Validates a digital X.509 certificate collected by the [Certificate Collector node](#).

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	<p>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.</p> <p>Default: Disabled</p>
Check Certificate Expiration	<p>When enabled, AM checks whether the certificate has expired.</p> <p>Default: Disabled</p>
Subject DN Attribute Used to Search LDAP for Certificates	<p>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.</p> <p>Default: CN</p>
Match Certificate to CRL	<p>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.</p> <p>Default: Disabled.</p>

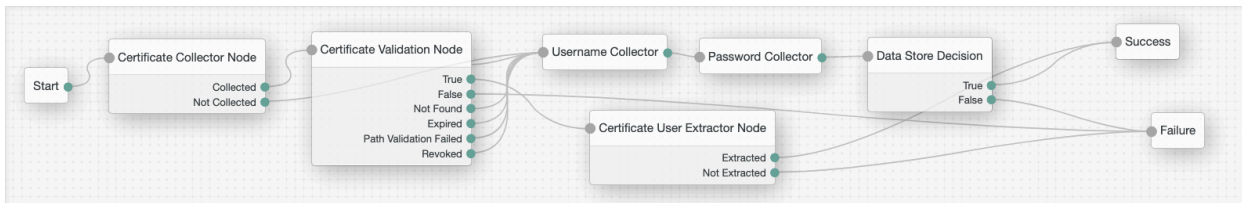
Property	Usage
<p>Issuer DN Attribute(s) Used to Search LDAP for CRLs</p>	<p>Specifies which attribute and value in the certificate Issuer DN AM uses to find the CRL in the LDAP directory.</p> <p>If only one attribute is specified, the LDAP search filter used is <code>(attr-name=attr-value-in-subject-DN)</code>.</p> <p>For example, if the subject DN of the issuer certificate is <code>C=US, CN=Some CA, serialNumber=123456</code>, and the attribute specified is <code>CN</code>, then the LDAP search filter used to find the CRL is <code>(CN=Some CA)</code>.</p> <p>Specify several CLR for the same CA issuer in a comma-separated list <code>(,)</code> where the names are in the same order as they occur in the subject DN.</p> <p>In this case, the LDAP search filter used is <code>(cn=attr1=attr1-value-in-subject-DN, attr2=attr2-value-in-subject-DN, ..., and so on.</code></p> <p>For example, if the subject DN of the issuer certificate is <code>C=US, CN=Some CA, serialNumber=123456</code>, and the attributes specified are <code>CN, serialNumber</code>, then the LDAP search filter used to find the CRL is <code>(cn=CN=Some CA, serialNumber=123456)</code>.</p> <p>Default: CN</p>
<p>HTTP Parameters for CRL Update</p>	<p>Specifies parameters that AM includes in any HTTP CRL call to the CA that issued the certificate.</p> <p>If the client or CA contains the Issuing Distribution Point Extension, AM uses this information to retrieve the CRL from the distribution point.</p> <p>Add the parameters as key pairs of values in a comma-separated list <code>(,)</code>. For example, <code>param1=value1, param2=value2</code>.</p>
<p>Cache CRLs in Memory</p>	<p>(LDAP distribution points only) When enabled, AM caches CRLs.</p> <p>Default: Enabled</p>

Property	Usage
Update CA CRLs from CRLDistributionPoint	<p>When enabled, AM updates the CRLs stored in the LDAP directory store if the CA certificate includes either the <code>IssuingDistributionPoint</code> or the <code>CRLDistributionPoint</code> extensions.</p> <p>Default: Enabled</p>
OCSP Validation	<p>When enabled, AM checks the revocation status of certificates using the Online Certificate Status Protocol (OCSP).</p> <p>The AM instance must have internet access, and you must configure OSCP for AM under Configure > Server Defaults > Security > Online Certificate Status Protocol Check.</p> <p>Default: Disabled</p>
LDAP Server Where Certificates are Stored	<p>Specifies the LDAP server that holds the certificates. Enter each server in the <code>ldap-server:port</code> format.</p> <p>AM servers can be associated with LDAP servers by writing multiple chains with the format <code>am_server ldapservice:port</code>. For example, <code>am.example.com ldap1.example.com:636</code>.</p> <p>To configure a secure connection, enable the Use SSL/TLS for LDAP Access property.</p>
LDAP Search Start or Base DN	<p>Valid base DN for the LDAP search, such as <code>dc=example,dc=com</code>. To associate AM servers with different search base DN, use the format <code>am_server base_dn</code>. For example, <code>am.example.com dc=example,dc=com</code> or <code>openam1.test.com dc=test,dc=com</code>.</p>
LDAP Server Authentication User	<p>Specifies the DN of the service account that AM uses to authenticate to the LDAP directory that holds the certificates. For example, <code>cn=LDAP User</code>.</p> <p>Default: <code>cn=Directory Manager</code></p>
LDAP Server Authentication Password	<p>Specifies the password of the user configured in the LDAP Server Authentication User property.</p>

Property	Usage
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. 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.

Outcomes

- True
- False

Properties

Property	Usage
Name of Cookie (<i>required</i>)	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.

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 <i>(required)</i>	Specify the latitude and longitude of at least one trusted location. Separate the values with a comma; for example, <code>37.7910855, -122.3951663</code> .
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.

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.

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.
Use Custom Matching Script	<p>Specifies whether to use a custom script to compare the collected metadata with saved device profiles.</p> <p>The script type must be <code>Decision node script</code> for authentication trees.</p> <div style="border: 1px solid #00aaff; padding: 5px; margin: 10px 0;"> <p>NOTE</p> <p>When a custom matching script is used, the <code>Acceptable Variance</code> and <code>Expiration</code> properties are ignored.</p> </div> <p>Default: <code>Authentication Tree Decision Node Script</code></p>
Custom Matching Script	<p>Specifies the custom script to use if the Use Custom Matching Script property is enabled.</p> <p>Only scripts of type <code>Decision node script</code> for authentication trees appear in the list.</p>

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 ForgeRock SDKs, the node can collect the following:

Device Metadata

Information such as the platform, versions, device name, hardware information, and the brand of the device being used.

The captured data is in JSON format, and stored in the authentication shared state in a variable named `forgeRock.device.profile`.

Device Location

Provides the last known latitude and longitude of the device's location.

The captured data is in JSON format, and stored in the authentication shared state in a variable named `forgeRock.device.location`.

The collection of geographical information requires end-user approval. A browser function drives this process. A pop-up displays, prompting for access to share the geographical location. The browser connection must be secure.

IMPORTANT

It is up to you what information you collect from users and devices.

Always use data responsibly and provide your users with appropriate control over data they share with you.

You are responsible for complying with any regulations or data protection laws.

In addition to the collected metadata, an `identifier` string in the JSON uniquely identifies the device.

Use this node with the [Device Profile Save node](#) to create a trusted profile from the collected data. You can use the trusted device profile in subsequent authentication attempts; for example, with the [Device Match node](#) and [Device Location Match node](#).

Outcomes

Single outcome path.

Properties

Property	Usage
Maximum Profile Size (KB)	<p>Specifies the maximum accepted size, in kilobytes, of a device profile.</p> <p>If the collected profile data exceeds this size, authentication fails.</p> <p>Default: 3</p>
Collect Device Metadata	Specifies whether device metadata is requested.
Collect Device Location	Specifies whether device location is requested.
Message	<p>Specifies an optional message to display to the user while the node collects the requested data.</p> <p>You can provide the message in multiple languages by specifying the locale in the <code>KEY</code> field; for example, <code>en-US</code>.</p> <p>The locale selected for display is based on the user's locale settings in their browser.</p> <p>Messages provided in the node override the defaults provided by AM.</p>

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.

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**.

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 it 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.

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	<p>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 .</p> <p>The higher the score returned from the device, the more likely the device is jailbroken, rooted, or is a potential security risk.</p> <p>Emulators score the maximum; 1 .</p>

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.

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.

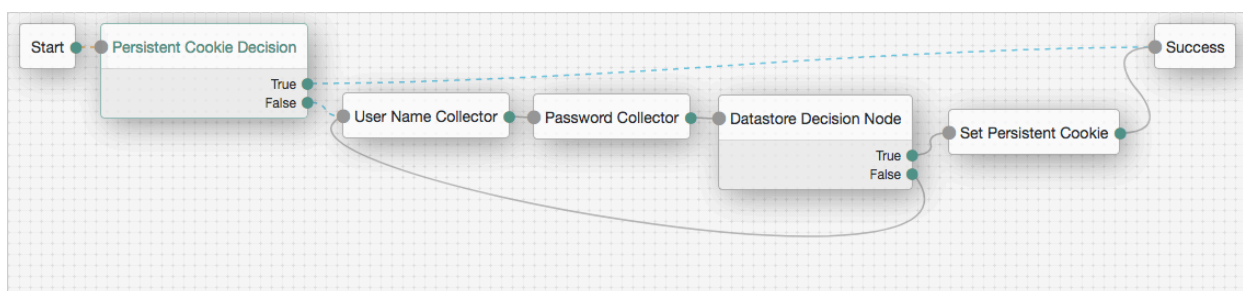
Properties

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 <code>Secure</code> flag to the persistent cookie. If the <code>Secure</code> 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 <code>HttpOnly</code> flag to the persistent cookie. When the <code>HttpOnly</code> flag is included, that cookie will not be accessible through JavaScript. According to RFC 6265  , the <code>HttpOnly</code> 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 <i>(required)</i>	<p>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.</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"> <p>IMPORTANT</p> <p>To consume the persistent cookies generated by the Set Persistent Cookie node, ensure they are using the same HMAC signing key.</p> </div> <p>To generate an HMAC signing key, run one of the following commands:</p> <pre style="background-color: #f9f9f9; padding: 10px; border: 1px solid #ccc;">\$ openssl rand -base64 32</pre> <p>or</p> <pre style="background-color: #f9f9f9; padding: 10px; border: 1px solid #ccc;">\$ cat /dev/urandom LC_ALL=C tr -dc 'a-zA-Z0-9' fold -w 32 head -n 1 base64</pre>
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

Store an additional custom cookie on the client.

This node uses the specified properties to create a cookie with a custom name and value, and optionally, sets attributes such as the cookie path, domain, expiry, and security flags.

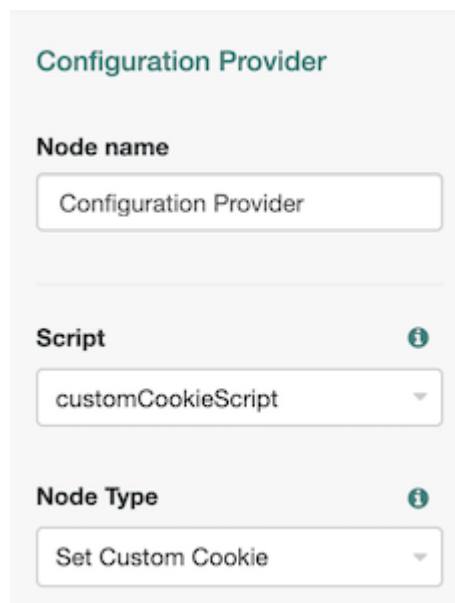
You can use this node with the [Configuration Provider node](#) to extend custom capabilities. For instance, create a `Config Provider` script to set custom static values

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:



The screenshot shows the configuration interface for a 'Configuration Provider' node. It features three main sections, each with a title and an information icon (i):


- Node name:** A text input field containing the value 'Configuration Provider'.
- Script:** A dropdown menu with 'customCookieScript' selected.
- Node Type:** A dropdown menu with 'Set Custom Cookie' selected.

Outcomes

Single outcome path.

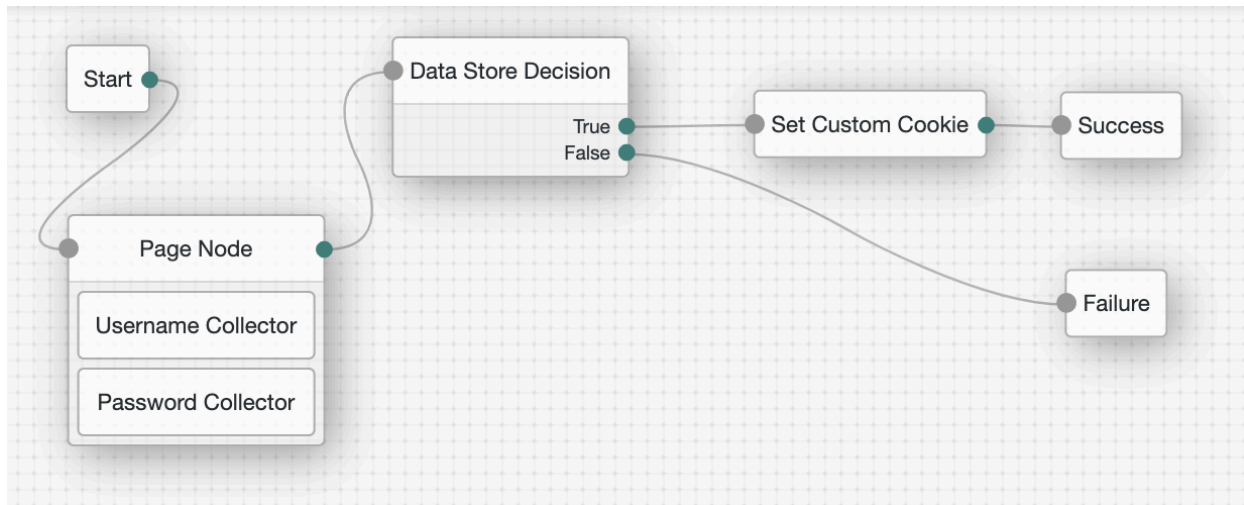
The cookie is created when AM next returns to the client.

Properties

Property	Usage
Custom Cookie Name <i>(required)</i>	<p>Sets the name of the custom cookie.</p> <p>The cookie name can contain any US-ASCII characters except for: space, tab, control, or a separator character (()<>@, ; : " / [] ? = \ { }).</p>
Custom Cookie Value <i>(required)</i>	Sets the value of the custom cookie.
Max Age	<p>Specifies the length of time the custom cookie remains valid, in seconds. If that time is exceeded, the cookie is no longer valid.</p> <p>Both the Max-Age and Expires attributes are set in the cookie to increase compatibility with different browsers.</p> <p>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.</p>
Custom Cookie Domain	Sets the domain that the custom cookie will be sent to.
Custom Cookie Path	Sets the path of the custom cookie.
Use Secure Cookie	<p>When enabled, adds the Secure flag to the custom cookie.</p> <p>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.</p>
Use HTTP Only Cookie	<p>When enabled, adds the HttpOnly flag to the custom cookie.</p> <p>When the HttpOnly flag is included, the cookie is not accessible to scripts.</p>
Custom Cookie SameSite attribute	<p>Sets the SameSite attribute of the custom cookie.</p> <p>The default value is LAX , to align with most modern browsers.</p> <p>Learn more in SameSite cookie rules.</p>

Example

This example uses this node in a login flow. The custom cookie is set 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.

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.

Property	Usage
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	<p>When enabled, adds the <code>Secure</code> flag to the persistent cookie.</p> <p>If the <code>Secure</code> 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.</p>
Use HTTP Only Cookie	<p>When enabled, adds the <code>HttpOnly</code> flag to the persistent cookie.</p> <p>When the <code>HttpOnly</code> flag is included, that cookie will not be accessible through JavaScript. According to RFC 6265 [↗], the <code>HttpOnly</code> 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)."</p>
HMAC Signing Key <i>(required)</i>	<p>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.</p> <div data-bbox="608 1294 1401 1498" style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p>IMPORTANT</p> <p>To consume the persistent cookies this node generates, ensure the nodes use the same HMAC signing key.</p> </div> <p>To generate an HMAC signing key, run one of the following commands:</p> <div data-bbox="608 1653 1401 1742" style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <pre>\$ openssl rand -base64 32</pre> </div> <p>or</p> <div data-bbox="608 1854 1401 1989" style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <pre>\$ cat /dev/urandom LC_ALL=C tr -dc 'a-zA-Z0-9' fold -w 32 head -n 1 base64</pre> </div>
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.

For information about the legacy/deprecated social authentication node and module implementations, refer to [Social authentication](#) in the *ForgeRock Access Management 7 Authentication and Single Sign-On Guide*.

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 <i>(required)</i>	Specifies the <code>client_id</code> parameter as described in section 2.2 of The OAuth 2.0 Authorization Framework (RFC 6749) .
Client Secret <i>(required)</i>	Specifies the <code>client_secret</code> parameter as described in section 2.3 of The OAuth 2.0 Authorization Framework (RFC 6749) .
Authentication Endpoint URL <i>(required)</i>	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: <code>https://accounts.google.com/o/oauth2/v2/auth</code>

Property	Usage
Access Token Endpoint URL <i>(required)</i>	<p>Specifies the URL to the endpoint handling access tokens as described in section 3.2 of The OAuth 2.0 Authorization Framework (RFC 6749).[↗]</p> <p>Example: <code>https://www.googleapis.com/oauth2/v4/token</code></p>
User Profile Service URL <i>(required)</i>	<p>Specifies the user profile URL that returns profile information.</p> <p>Example: <code>https://www.googleapis.com/oauth2/v3/userinfo</code></p>
OAuth Scope <i>(required)</i>	<p>Specifies a list of user profile attributes that the client application requires, according to The OAuth 2.0 Authorization Framework (RFC 6749).[↗]</p> <p>Ensure you use the correct scope delimiter required by the identity provider, including commas or spaces.</p> <p>The list depends on the permissions that the resource owner, such as the end user, grants to the client application.</p>
Scope Delimiter <i>(required)</i>	<p>Specifies the delimiter used to separate scope values.</p> <p>Some authorization servers use non-standard separators for scopes, for example commas.</p>
Redirect URL <i>(required)</i>	<p>Specifies the URL the user is redirected to by the social identity provider after authenticating.</p> <p>For authentication trees in AM, set this property to the URL of the UI. For example, <code>https://openam.example.com:8443/openam/XUI/</code></p>
Social Provider <i>(required)</i>	<p>Specifies the name of the social provider for which this module is being set up.</p> <p>Example: Google</p>
Auth ID Key <i>(required)</i>	<p>Specifies the attribute the social identity provider uses to identify an authenticated individual.</p> <p>Example: id</p>

Property	Usage
Use Basic Auth	<p>Specifies that the client uses HTTP Basic authentication when authenticating to the social provider.</p> <p>Default: <code>true</code></p>
Account Provider <i>(required)</i>	<p>Specifies the name of the class that implements the account provider.</p> <p>Default: <code>org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider</code></p>
Account Mapper <i>(required)</i>	<p>Specifies the name of the class that implements the method of locating local accounts based on the attributes returned from the social identity provider.</p> <p>Provided implementations are:</p> <p><code>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper</code></p> <p>The Account Mapper classes can take two constructor parameters:</p> <ol style="list-style-type: none"> 1. A comma-separated list of attributes 2. A prefix to apply to their values. <p>For example, to prefix all received property values with <code>facebook-</code> before searching, specify:</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <pre>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper * facebook-</pre> </div>

Property	Usage
<p>Attribute Mapper (<i>required</i>)</p>	<p>Specifies the list of fully qualified class names for implementations that map attributes from the OAuth 2.0 authorization server to AM profile attributes.</p> <p>Provided implementations are:</p> <pre>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper</pre> <p>The Attribute Mapper classes can take two constructor parameters to help differentiate between the providers:</p> <ol style="list-style-type: none"> 1. A comma-separated list of attributes 2. A prefix to apply to their values. <p>For example, to prefix all incoming values with facebook- , specify:</p> <pre>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper * facebook-</pre> <p>To prefix all incoming values use an asterisk (*) as the attribute list. This prefixes all values, including email addresses, postal addresses, and so on.</p>

Property	Usage
Account Mapper Configuration	<p data-bbox="608 197 1350 322">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.</p> <p data-bbox="608 365 1350 443">Valid values are in the form <i>provider-attr=local-attr</i>.</p> <p data-bbox="608 486 751 521">Examples:</p> <div data-bbox="608 555 1398 696" style="border: 1px solid #ccc; padding: 5px;"><pre data-bbox="632 589 884 667">email=mail id=facebook-id</pre></div> <p data-bbox="639 723 687 757">TIP</p> <p data-bbox="639 779 1358 994">When using the <code>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper</code> class, you can parse JSON objects in mappings using dot notation.</p> <p data-bbox="639 1037 1174 1072">For example, given a JSON payload of:</p> <div data-bbox="639 1106 1362 1417" style="border: 1px solid #ccc; padding: 5px;"><pre data-bbox="663 1140 1054 1395">{ "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } }</pre></div> <p data-bbox="639 1458 1114 1536">You can create a mapper, such as <code>name.first_name=cn</code>.</p>

Property	Usage
Attribute Mapper Configuration	<p>Map of OAuth 2.0 provider user account attributes to local user profile attributes, with values in the form <i>provider-attr=local-attr</i>.</p> <p>Examples:</p> <pre> first_name=givenname last_name=sn name=cn email=mail id=facebook-id first_name=facebook-fname last_name=facebook-lname email=facebook-email </pre> <p>TIP</p> <p>When using the <code>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper</code> class, you can parse JSON objects in mappings using dot notation.</p> <p>For example, given a JSON payload of:</p> <pre> { "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } } </pre> <p>You can create a mapper, such as <code>name.first_name=cn</code>.</p>
Save attributes in the session	When enabled, saves the attributes in the Attribute Mapper Configuration field to the AM session.

Property	Usage
OAuth 2.0 Mix-Up Mitigation Enabled	<p>Controls whether the OAuth 2.0 authentication node carries out additional verification steps when it receives the authorization code from the authorization server.</p> <p>Specifies that the client must compare the issuer identifier of the authorization server upon registration with the issuer value returned as the <code>iss</code> 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 <code>client_id</code> response parameter.</p> <p>When this is enabled, set the Token Issuer property so that the validation can succeed. The authorization code response contains an issuer value (<code>iss</code>) for the client to validate.</p> <div data-bbox="608 958 1401 1128" style="border: 1px solid #0070C0; padding: 5px;"> <p>NOTE</p> <p>Refer to the authorization server's documentation for the value it uses for the issuer field.</p> </div> <p>For more information, refer to section 4 of OAuth 2.0 Mix-Up Mitigation Draft.</p>
Token Issuer	<p>Corresponds to the expected issuer identifier value in the <code>iss</code> field of the ID token.</p> <p>Example: <code>https://accounts.google.com</code></p>

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.

For information about the legacy/deprecated social authentication node and module implementations, refer to [Social authentication](#) in the *ForgeRock Access Management 7 Authentication and Single Sign-On Guide*.

The OpenID Connect node implements the [Authorization code grant](#).

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 <i>(required)</i>	Specifies the <code>client_id</code> parameter as described in section 2.2 of The OAuth 2.0 Authorization Framework (RFC 6749) . ^[↗]
Client Secret <i>(required)</i>	Specifies the <code>client_secret</code> parameter as described in section 2.3 of The OAuth 2.0 Authorization Framework (RFC 6749) . ^[↗]
Authentication Endpoint URL <i>(required)</i>	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: <code>https://accounts.google.com/o/oauth2/v2/auth</code>
Access Token Endpoint URL <i>(required)</i>	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: <code>https://www.googleapis.com/oauth2/v4/token</code>
User Profile Service URL <i>(required)</i>	Specifies the user profile URL that returns profile information. If not specified, attributes are mapped from the claims returned by the <code>id_token</code> , and no call to a user profile endpoint is made. Example: <code>https://www.googleapis.com/oauth2/v3/userinfo</code>

Property	Usage
OAuth Scope	<p>Specifies a list of user profile attributes that the client application requires, according to The OAuth 2.0 Authorization Framework (RFC 6749).[↗]</p> <p>Ensure you use the correct scope delimiter required by the identity provider, including commas or spaces.</p> <p>The list depends on the permissions that the resource owner, such as the end user, grants to the client application.</p>
Redirect URL	<p>Specifies the URL the user is redirected to by the social identity provider after authenticating.</p> <p>For authentication trees in AM, set this property to the URL of the UI. For example, <code>https://openam.example.com:8443/openam/XUI/</code>.</p>
Social Provider <i>(required)</i>	<p>Specifies the name of the OpenID Connect provider for which this node is being set up.</p> <p>Example: Google</p>
Auth ID Key	<p>Specifies the attribute the social identity provider uses to identify an authenticated individual.</p> <p>Example: sub</p>
Use Basic Auth	<p>Specifies that the client uses HTTP Basic authentication when authenticating to the social provider.</p> <p>Default: true</p>
Account Provider	<p>Specifies the name of the class that implements the account provider.</p> <p>Default: <code>org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider</code></p>

Property	Usage
Account Mapper	<p>Specifies the name of the class that implements the method of locating local accounts based on the attributes returned from the social identity provider.</p> <p>The provided implementations is <code>org.forgerock.openam.authentication.modules.oidc.JwtAttributeMapper</code> .</p> <p>The Account Mapper classes can take two constructor parameters:</p> <ol style="list-style-type: none"> 1. A comma-separated list of attributes 2. A prefix to apply to their values. <p>For example, to prefix all received property values with <code>openid-</code> before searching, specify:</p> <pre>org.forgerock.openam.authentication.modules.oidc.JwtAttributeMapper * openid-</pre>
Attribute Mapper	<p>Specifies the list of fully qualified class names for implementations that map attributes from the authorization server to AM profile attributes.</p> <p>The provided implementations is <code>org.forgerock.openam.authentication.modules.oidc.JwtAttributeMapper</code> .</p> <p>The Attribute Mapper classes can take two constructor parameters to help differentiate between the providers:</p> <ol style="list-style-type: none"> 1. A comma-separated list of attributes 2. A prefix to apply to their values. <p>For example, to prefix incoming <code>iplanet-am-user-alias-list</code> values with <code>openid-</code> , specify:</p> <pre>org.forgerock.openam.authentication.modules.oidc.JwtAttributeMapper</pre>

Property	Usage
iplanet-am-user-alias-list	<p>openid-</p> <p>To prefix all incoming values use an asterisk (*) as the attribute list. This prefixes all values, including email addresses, postal addresses, and so on.</p>
Account Mapper Configuration	<p>Specifies the attribute configuration used to map the account of the user authenticated in the provider to the local identity store in AM.</p> <p>To add a mapping, specify the name of the provider attribute as the key, and the local attribute to map to as the value.</p> <p>For example, click Add, then specify <code>sub</code> in the Key field and <code>iplanet-am-user-alias-list</code> in the Value field, and click +.</p>
Attribute Mapper Configuration	<p>Specifies how to map provider user attributes to local user profile attributes.</p> <p>To add a mapping, specify the name of the provider attribute as the Key, and the local attribute to map to as the Value.</p> <p>For example, click Add, then specify <code>id</code> in the Key field and <code>facebook-id</code> in the Value field, and click +.</p> <p>Examples:</p> <pre data-bbox="608 1429 1398 1832"> first_name=givenname last_name=sn name=cn email=mail id=facebook-id first_name=facebook-fname last_name=facebook-lname email=facebook-email </pre>
Save attributes in the session	<p>When enabled, saves the attributes in the Attribute Mapper Configuration field to the AM session.</p>

Property	Usage
OAuth 2.0 Mix-Up Mitigation Enabled	<p>Controls whether the authentication node carries out additional verification steps when it receives the authorization code from the authorization server.</p> <p>Specifies that the client must compare the issuer identifier of the authorization server upon registration with the issuer value returned as the <code>iss</code> 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 <code>client_id</code> response parameter.</p> <p>When this is enabled, set the Token Issuer property so that the validation can succeed. The authorization code response contains an issuer value (<code>iss</code>) for the client to validate.</p> <div data-bbox="608 958 1401 1128" style="border: 1px solid #0070C0; padding: 5px;"> <p>NOTE</p> <p>Refer to the authorization server's documentation for the value it uses for the issuer field.</p> </div> <p>For more information, refer to section 4 of OAuth 2.0 Mix-Up Mitigation Draft [↗].</p>
Token Issuer (<i>required</i>)	<p>Corresponds to the expected issuer identifier value in the <code>iss</code> field of the ID token.</p> <p>Example: <code>https://accounts.google.com</code></p>

Property	Usage
OpenID Connect Validation Type <i>(required)</i>	<p>Specifies how to validate the ID token received from the OpenID Connect provider.</p> <p>This ignores keys specified in JWT headers, such as <code>jku</code> and <code>jwe</code>.</p> <p>The following options are available to validate an incoming OpenID Connect ID token:</p> <p><i>Well Known URL (Default)</i> Retrieves the provider's keys based on the information provided in its OpenID Connect configuration URL.</p> <p>Specify the provider's configuration URL in the OpenID Connect Validation Value field; for example, <code>https://accounts.google.com/.well-known/openid-configuration</code>.</p> <p><i>Client Secret</i> Validates the ID token signature with a specified client secret key.</p> <p>Specify the key to use in the OpenID Connect Validation Value field.</p> <p><i>JWK URL</i> Retrieve the necessary JSON web key from the URL that you specify.</p> <p>Specify the provider's JWK URI in the OpenID Connect Validation Value field; for example, <code>https://www.googleapis.com/oauth2/v3/certs</code>.</p>
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.

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.

Outcomes

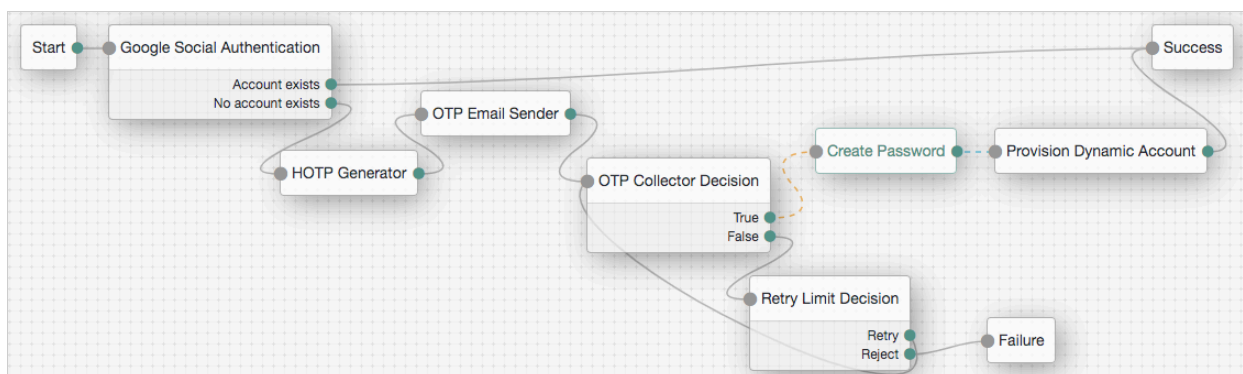
Single outcome path.

Properties

Property	Usage
Account Provider	<p>Specifies the name of the class that implements the account provider.</p> <p>Default: <code>org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider</code></p>

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.

For information about the legacy/deprecated social authentication node and module implementations, refer to [Social authentication](#) in the *ForgeRock Access Management 7 Authentication and Single Sign-On Guide*.

Ensure you have configured the details of the IDM instance in AM, by navigating to **Configure > Global Services > IDM Provisioning**.

Outcomes

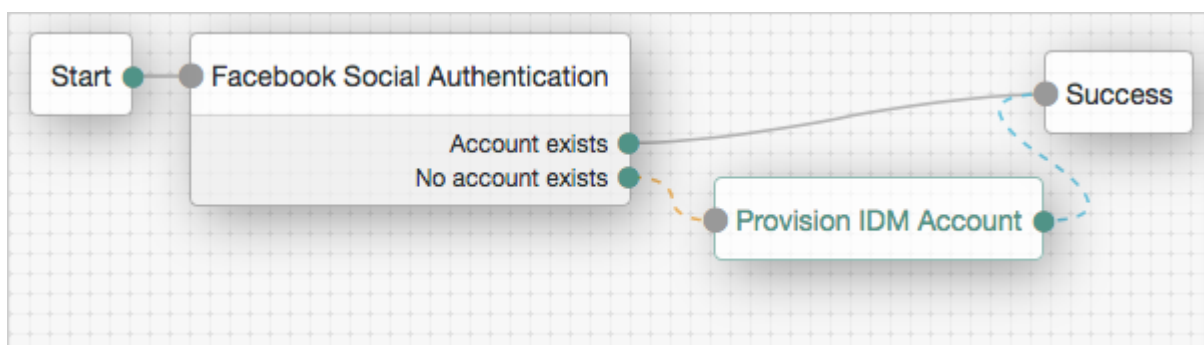
Single outcome path.

Properties

Property	Usage
Account Provider	Specifies the name of the class that implements the account provider. Default: <code>org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider</code>

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](#).


Outcomes

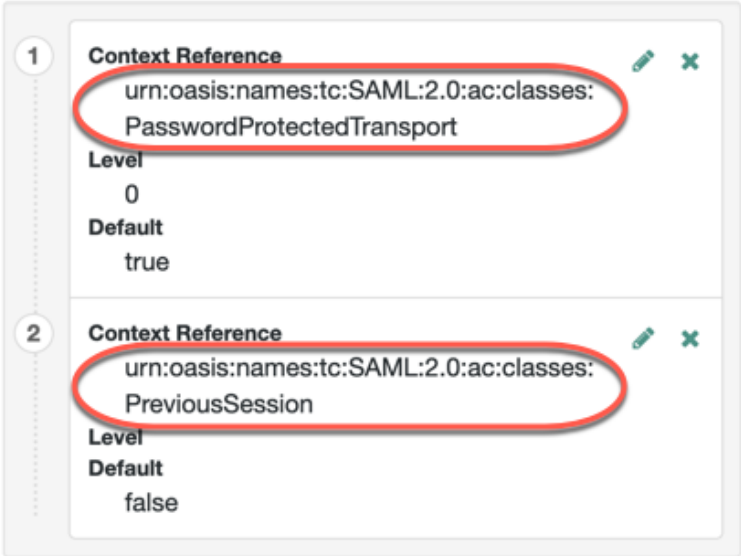
- `Account exists`
- `No account exists`

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 <code>/Realm Name/SP Name</code> .
Allow IdP to Create NameID	<p>Specifies whether the IdP should create a new identifier for the authenticating user if none exists.</p> <p>For detailed information, refer to the section on the <code>AllowCreate</code> property in SAML Version 2.0 Errata 05.</p> <p>Default: Enabled</p>

Property	Usage
Comparison Type	<p data-bbox="603 203 1246 280">Specifies a comparison method to evaluate authentication context classes or statements.</p> <p data-bbox="603 324 1382 584">The value specified in this property overrides the value set in the SP configuration in AM admin UI under Realms > <i>Realm Name</i> > Applications > Federation > Entity Providers > <i>Service Provider Name</i> > Assertion Content > Authentication Context > Comparison Type.</p> <p data-bbox="603 629 1305 705">Valid comparison methods are <code>exact</code>, <code>minimum</code>, <code>maximum</code>, or <code>better</code>.</p> <p data-bbox="603 750 1386 920">For more information about the comparison methods, refer to the section on the <code><RequestedAuthnContext></code> element in Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 .</p> <p data-bbox="603 965 855 987">Default: <code>minimum</code></p>

Property	Usage
Authentication Context Class Reference	<p>(Optional) Specifies one or more URIs for authentication context classes to be included in the SAML request.</p> <p>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.</p> <p>Any authentication context class you specify in this field must be supported for the service provider. In the AM admin UI, go to Realms > <i>Realm Name</i> > Applications > Federation > Entity Providers > <i>Service Provider Name</i> > Assertion Content > Authentication Context.</p>  <p>When specifying multiple authentication context classes, use the <code> </code> character to separate the classes. For example:</p> <pre>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport urn:oasis:names:tc:SAML:2.0:ac:classes:TimesyncToken</pre>

Property	Usage
Authentication Context Declaration Reference	<p>(Optional) Specifies one or more URIs that identify authentication context declarations.</p> <p>When specifying multiple URIs, use the <code> </code> character to separate the URIs.</p> <p>For more information, refer to the section on the <code><RequestedAuthnContext></code> element in Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0.</p>
Request Binding	<p>Specifies the format the SP will use to send the authentication request to the IdP.</p> <p>Valid values are <code>HTTP-Redirect</code> and <code>HTTP-POST</code>.</p> <p>Default: <code>HTTP-Redirect</code></p>
Response Binding	<p>Specifies the format the IdP will use to send the response to the SP.</p> <p>Valid values are <code>HTTP-POST</code> and <code>HTTP-Artifact</code>.</p> <p>Default: <code>HTTP-Artifact</code></p>
Force IdP Authentication	<p>Specifies whether the IdP forces authentication or if it can reuse existing security contexts.</p> <p>Default: <code>Disabled</code></p>
Passive Authentication	<p>Specifies whether the IdP uses passive authentication or not.</p> <p>Passive authentication requires the IDP to only use authentication methods that do not require user interaction; for example, authenticating using an X.509 certificate.</p> <p>Default: <code>Disabled</code></p>

Property	Usage
NameID Format	<p>Specifies the SAML name ID format that will be requested in the SAML authentication request. For example:</p> <pre>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</pre> <p>Default: urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</p>

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.

For information about the legacy/deprecated social authentication node and module implementations, refer to [Social authentication](#) in the *ForgeRock Access Management 7 Authentication and Single Sign-On Guide*.

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 <code>client_id</code> parameter as provided by Facebook.
Client Secret	Specifies the <code>client_secret</code> 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: <code>https://www.facebook.com/dialog/oauth</code>
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: <code>https://graph.facebook.com/v2.12/oauth/access_token</code>
User Profile Service URL	Specifies the user profile URL that returns profile information. Default: <code>https://graph.facebook.com/v2.6/me?fields=name%2Cemail%2Cfirst_name%2Clast_name</code>
OAuth Scope	Specifies a comma-separated list of user profile attributes the client application requires, according to The OAuth 2.0 Authorization Framework (RFC 6749) . The list depends on the permissions the resource owner, such as the end user, grants to the client application.

Property	Usage
Redirect URL	<p>Specifies the URL the user is redirected to by Facebook after authenticating to continue the flow.</p> <p>Set this property to the URL of the AM UI. For example, <code>https://openam.example.com:8443/openam/XUI/</code>.</p> <div style="border: 1px solid green; padding: 5px; margin-top: 10px;"> <p>TIP</p> <p>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, <code>https://openam.example.com:8443/openam/XUI/?realm=/mySubRealm</code>.</p> <p>For more information, refer to Configure DNS aliases to access a realm.</p> </div>
Social Provider	<p>Specifies the name of the social provider for which this node is being set up.</p> <p>Default: facebook</p>
Auth ID Key	<p>Specifies the attribute the social identity provider uses to identify an authenticated individual.</p> <p>Default: id</p>
Use Basic Auth	<p>Specifies that the client uses HTTP Basic authentication when authenticating to the social provider.</p> <p>Default: true</p>
Account Provider	<p>Specifies the name of the class that implements the account provider.</p> <p>Default: <code>org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider</code></p>

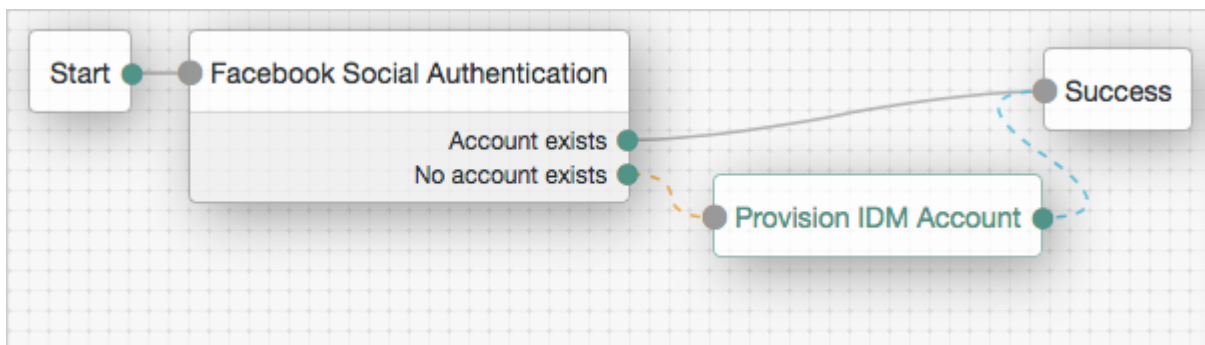
Property	Usage
Account Mapper	<p>Specifies the name of the class that implements the method of locating local accounts based on the attributes returned from Facebook.</p> <p>Default: <code>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper</code></p>
Attribute Mapper	<p>Specifies the list of fully qualified class names for implementations that map attributes from Facebook to AM profile attributes.</p> <p>Default: <code>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper uid facebook-</code></p>
Account Mapper Configuration	<p>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 <i>provider-attr=local-attr</i>.</p> <p>Default: <code>id=uid</code>.</p> <div data-bbox="608 1189 1401 2033" style="border: 1px solid green; padding: 10px;"> <p>TIP</p> <p>When using the <code>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper</code> class, you can parse JSON objects in mappings using dot notation.</p> <p>For example, given a JSON payload of:</p> <pre style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">{ "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } }</pre> <p>You can create a mapper, such as <code>name.first_name=cn</code>.</p> </div>

Property	Usage
Attribute Mapper Configuration	<p>Map of Facebook user account attributes to local user profile attributes, with values in the form <i>provider-attr=local-attr</i>.</p> <p>Default: name=cn, last_name=sn, id=uid, first_name=givenname, email=mail.</p> <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p>TIP</p> <p>When using the <code>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper</code> class, you can parse JSON objects in mappings using dot notation.</p> <p>For example, given a JSON payload of:</p> <pre style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">{ "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } }</pre> <p>You can create a mapper, such as <code>name.first_name=cn</code>.</p> </div>
Save attributes in the session	<p>When enabled, saves the attributes in the Attribute Mapper Configuration field to the AM session.</p> <p>Default: true.</p>

Property	Usage
OAuth 2.0 Mix-Up Mitigation Enabled	<p>Controls whether the authentication node carries out additional verification steps when it receives the authorization code from the authorization server.</p> <p>Specifies that the client must compare the issuer identifier of the authorization server upon registration with the issuer value returned as the <code>iss</code> 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 <code>client_id</code> response parameter.</p> <p>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 (<code>iss</code>) for the client to validate.</p> <p>For more information, refer to section 4 of OAuth 2.0 Mix-Up Mitigation Draft ↗.</p>
Token Issuer	<p>Corresponds to the expected issuer identifier value in the <code>iss</code> field of the ID token.</p> <p>Example: <code>https://graph.facebook.com</code></p>

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

NOTE

This node and its related services, are deprecated.

For information about the legacy/deprecated social authentication node and module implementations, refer to [Social authentication](#) in the *ForgeRock Access Management 7 Authentication and Single Sign-On Guide*.

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 (<i>required</i>)	Specifies the <code>client_id</code> parameter as provided by Google.
Client Secret (<i>required</i>)	Specifies the <code>client_secret</code> parameter as provided by Google.
Authentication Endpoint URL	Specifies the URL to the social provider's endpoint handling authentication as described in section 3.1 of The OAuth 2.0 Authorization Framework (RFC 6749) . Default: <code>https://accounts.google.com/o/oauth2/v2/auth</code>
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: <code>https://www.googleapis.com/oauth2/v4/token</code>
User Profile Service URL	Specifies the user profile URL that returns profile information. Default: <code>https://www.googleapis.com/oauth2/v3/userinfo</code>

Property	Usage
OAuth Scope	<p>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.</p> <p>Default: <code>profile email</code>.</p>
Redirect URL	<p>Specifies the URL the user is redirected to by Google after authenticating to continue the flow.</p> <p>Set this property to the URL of the AM UI. For example, <code>https://openam.example.com:8443/openam/XUI/</code>.</p> <div data-bbox="608 779 1401 1238" style="border: 1px solid green; padding: 10px;"> <p>TIP</p> <p>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, <code>https://openam.example.com:8443/openam/XUI/?realm=/mySubRealm</code>.</p> <p>For more information, refer to Configure DNS aliases to access a realm.</p> </div>
Social Provider	<p>Specifies the name of the social provider for which this node is being set up.</p> <p>Default: <code>google</code></p>
Auth ID Key	<p>Specifies the attribute the social identity provider uses to identify an authenticated individual.</p> <p>Default: <code>sub</code></p>
Use Basic Auth	<p>Specifies that the client uses HTTP Basic authentication when authenticating to Google.</p> <p>Default: <code>true</code></p>

Property	Usage
Account Provider	<p>Specifies the name of the class that implements the account provider.</p> <p>Default: <code>org.forgerock.openam.authentication.modules.common.mapping.DefaultAccountProvider</code></p>
Account Mapper	<p>Specifies the name of the class that implements the method of locating local accounts based on the attributes returned from Google.</p> <p>Default: <code>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper</code></p>
Attribute Mapper	<p>Specifies the list of fully qualified class names for implementations that map attributes from Google to AM profile attributes.</p> <p>Default: <code>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper iplanet-am-user-alias-list google-</code></p>

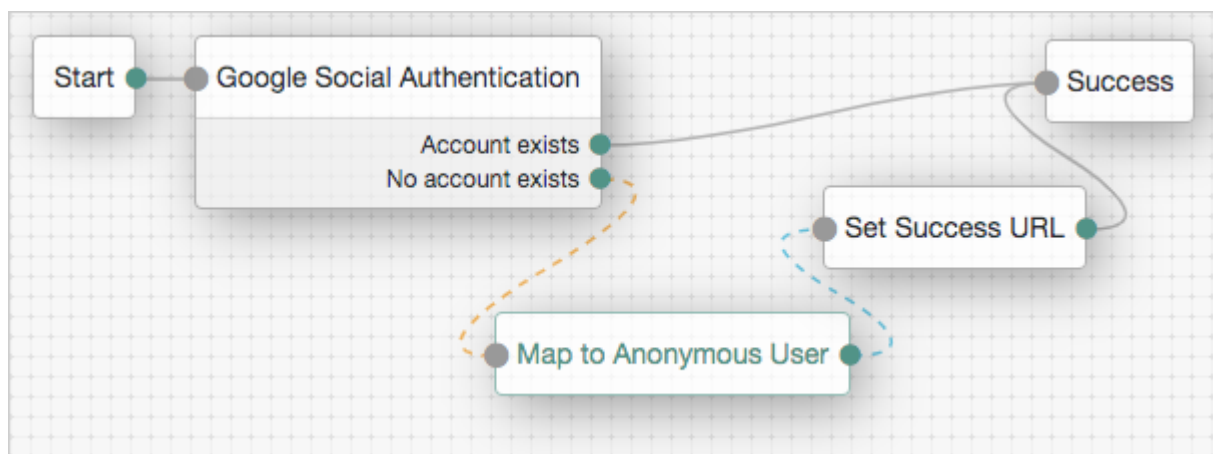
Property	Usage
Account Mapper Configuration	<p data-bbox="608 203 1398 367">Specifies the attribute configuration used to map the account of the user authenticated in the Social Google provider to the local data store in AM. Valid values are in the form <i>provider-attr=local-attr</i>.</p> <p data-bbox="608 416 868 443">Default: <code>sub=uid</code>.</p> <div data-bbox="608 472 1398 1328" style="border: 1px solid green; padding: 10px;"><p data-bbox="643 483 687 510">TIP</p><p data-bbox="643 539 1358 748">When using the <code>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper</code> class, you can parse JSON objects in mappings using dot notation.</p><p data-bbox="643 797 1174 824">For example, given a JSON payload of:</p><pre data-bbox="667 891 1054 1144" style="border: 1px solid #ccc; padding: 10px;">{ "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } }</pre></div> <p data-bbox="643 1216 1114 1288">You can create a mapper, such as <code>name.first_name=cn</code>.</p>

Property	Usage
Attribute Mapper Configuration	<p>Map of Google user account attributes to local user profile attributes, with values in the form <i>provider-attr=local-attr</i>.</p> <p>Default: sub=uid, name=cn, given_name=givenName, family_name=sn, email=mail.</p> <div style="border: 1px solid green; padding: 10px; margin: 10px 0;"> <p>TIP</p> <p>When using the <code>org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper</code> class, you can parse JSON objects in mappings using dot notation.</p> <p>For example, given a JSON payload of:</p> <pre style="background-color: #f9f9f9; padding: 10px; border: 1px solid #ccc;">{ "sub" : "12345", "name" : { "first_name" : "Demo", "last_name" : "User" } }</pre> <p>You can create a mapper, such as <code>name.first_name=cn</code>.</p> </div>
Save attributes in the session	<p>When enabled, saves the attributes in the Attribute Mapper Configuration field to the AM session.</p> <p>Default: true.</p>

Property	Usage
OAuth 2.0 Mix-Up Mitigation Enabled	<p>Controls whether the authentication node carries out additional verification steps when it receives the authorization code from the authorization server.</p> <p>Specifies that the client must compare the issuer identifier of the authorization server upon registration with the issuer value returned as the <code>iss</code> 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 <code>client_id</code> response parameter.</p> <p>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 (<code>iss</code>) for the client to validate.</p> <p>For more information, refer to section 4 of OAuth 2.0 Mix-Up Mitigation Draft ↗.</p>
Token Issuer	<p>Corresponds to the expected issuer identifier value in the <code>iss</code> field of the ID token.</p> <p>Example: <code>https://accounts.google.com</code></p>

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.

For information about the legacy/deprecated social authentication node and module implementations, refer to [Social authentication](#) in the *ForgeRock Access Management 7 Authentication and Single Sign-On Guide*.

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?
ForgeRock Identity Cloud	Yes
ForgeRock Access Management (self-managed)	Yes
ForgeRock Identity Platform (self-managed)	Yes

Inputs

This node reads the user's selected social identity provider from shared state.

Implement the [Select Identity Provider node](#) before this node to capture the social provider name.

Dependencies

- The Social Identity Provider service must be configured with the details of at least one social identity provider.
- The user must have selected a social identity provider in a previous node in the journey.

Configuration

Property	Usage
Transformation Script (required)	<p>This script is used after the configured provider's <i>normalization</i> script has mapped the social identity provider's attributes to a profile format compatible with AM. The <i>transformation</i> script then transforms a normalized social profile to an identity.</p> <p>Select <code>Normalized Profile to Identity</code>, or your own script that you have created to transform the profile to an identity object.</p> <p>To view the scripts and bindings, refer to normalized-profile-to-identity.js.</p> <p>Normalization scripts (<code><Identity provider>-profile-normalization.*</code>) are not suitable for this purpose.</p>
Username Attribute	ForgeRock Identity Platform deployments only.
Client Type	<p>Specify the client type you are using to authenticate to the provider.</p> <p>Use the default, <code>BROWSER</code>, with ForgeRock-provided user interfaces or the ForgeRock SDK for JavaScript. This causes the node to return the RedirectCallback.</p> <p>Select <code>NATIVE</code> with the ForgeRock SDKs for Android or iOS. This causes the node to return the IdPCallback.</p>

Outputs

- When the node attempts to authenticate the user through the social provider, it sets `expectProfileInformation` to `true` in the node state. If no profile

information is returned, the journey follows the `Social auth interrupted` outcome.

- If the node retrieves the profile information from the social identity provider, performs the required transformations, and locates a matching identity, it puts the identity into the node state to authenticate the user.

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.

Social auth interrupted

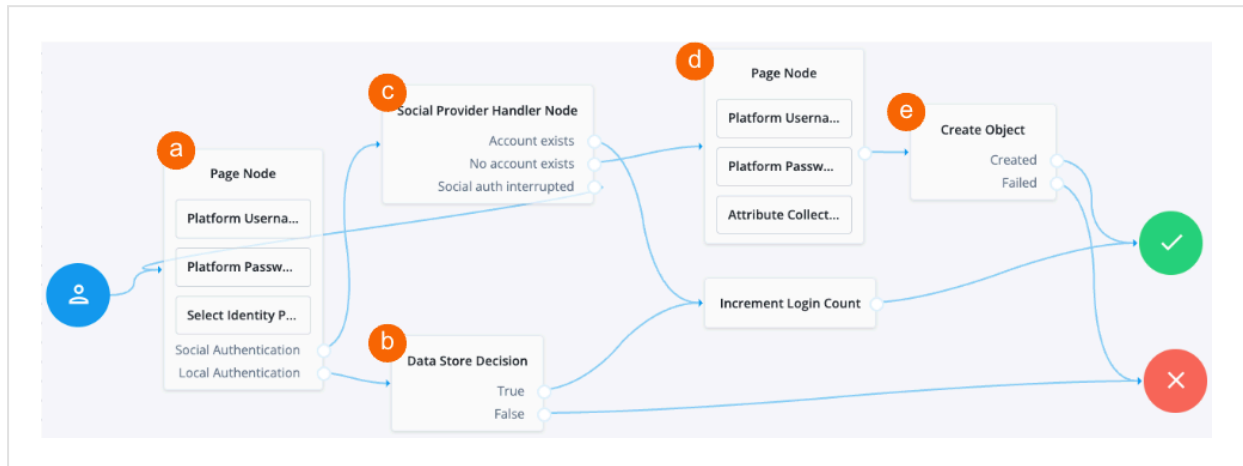
The user interrupted the social authentication journey after the node requested profile information from the social identity provider. This can happen in the following situations:

- 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 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 for authentication.
- 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 .

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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

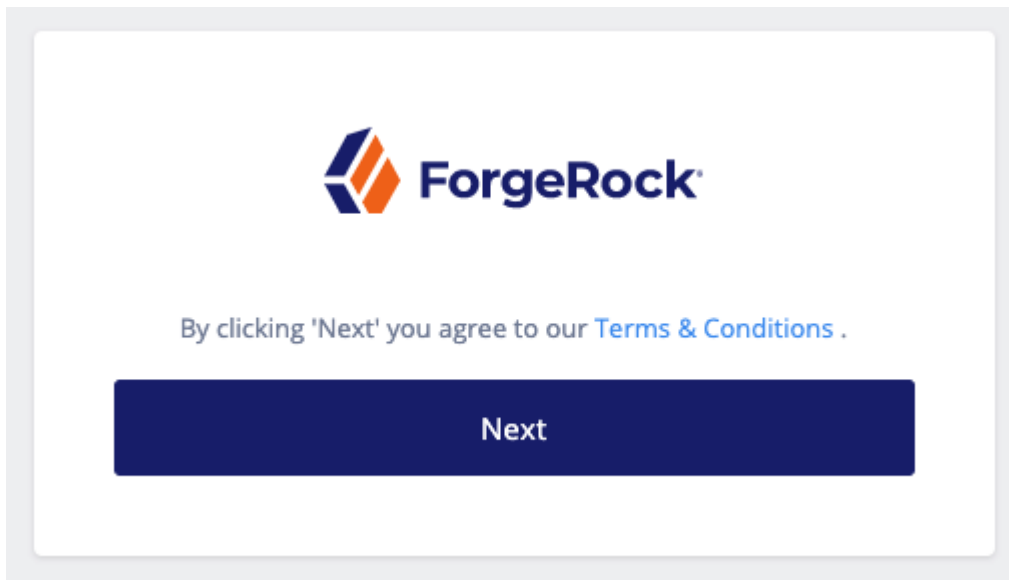
You set terms and conditions in the Identity Platform admin UI. For more information, refer to [Terms and conditions](#).

Use this node for registration, or combined with the [Terms and Conditions Decision node](#) for progressive profiling or log in.

Outcomes

Single outcome path.

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



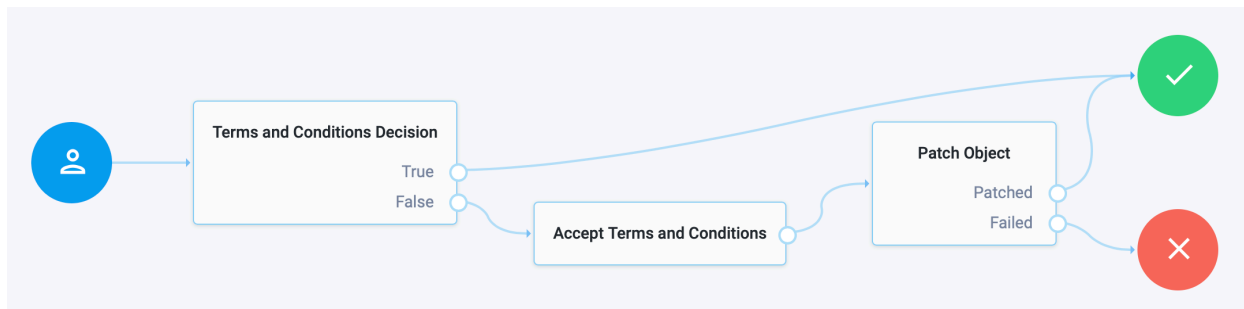
Properties

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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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](#).

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	<p>When enabled, validate the content against any policies specified in the IDM schema for each collected attribute.</p> <p>For more information, refer to Use policies to validate data in the IDM documentation.</p> <p>If you enable this property, the collected attributes must be <i>User Editable</i> in IDM. To make an attribute user-editable in the IDM admin UI:</p> <ol style="list-style-type: none"> 1. Go to Configure > Managed Objects > <i>object-name</i>. 2. Click the pencil (✎) icon then click Show advanced options. 3. Select the User Editable toggle. <p>For details, refer to Property Configuration Properties in the IDM documentation.</p>
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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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.

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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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` .

Properties

Property	Usage
Comparison Operation	<p>The operation to perform on the object attribute:</p> <p>PRESENT Checks the existence of an attribute regardless of its value.</p> <p>EQUALS Checks if the object's attribute value equals the configured comparison value.</p>
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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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.

Properties

Property	Usage
All Mappings Required	If enabled, all mappings listed by this node require consent in order to move forward.

Property	Usage
Privacy & Consent Message	Localized message providing the privacy and consent notice. The key is the language, such as <code>en</code> or <code>fr</code> , 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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

Any managed object attributes that are marked as required in IDM must be collected during authentication in order to create the new object.

Properties

Property	Usage
Identity Resource	<p>The type of IDM managed identity resource object that this node creates. It must match the identity resource type for the current flow.</p> <div data-bbox="842 1391 906 1424" data-label="Section-Header"> <p>TIP</p> </div> <p>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.</p> <p>Identity managed object types are preceded by the  icon.</p>

Create Password node

Lets users create a password when provisioning an account.

NOTE

NOTE

This node and its related services, are deprecated.

For information about the legacy/deprecated social authentication node and module implementations, refer to [Social authentication](#) in the *ForgeRock Access Management 7 Authentication and Single Sign-On Guide*.

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.

Outcomes

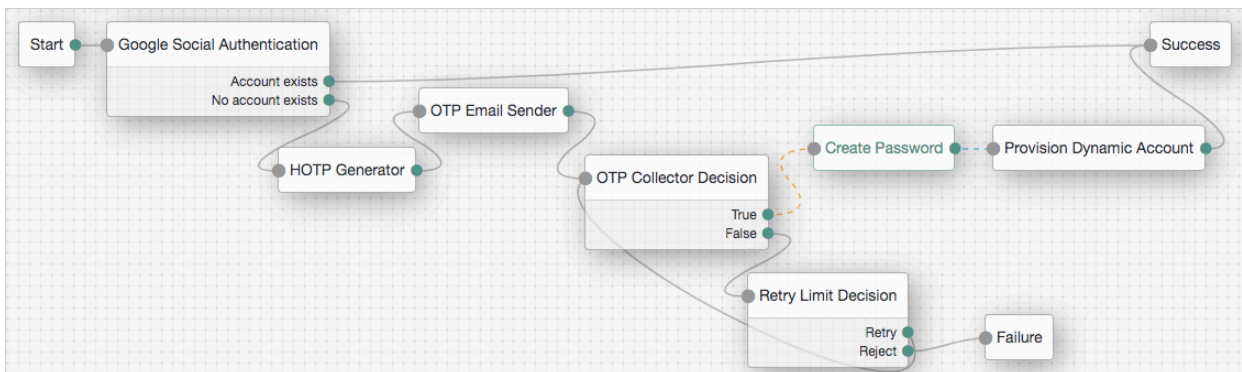
Single outcome path.

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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

To email the username to the user instead, use the [Identify Existing User node](#) combined with a [Email Suspend node](#) or [Email Template node](#).

Properties

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

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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

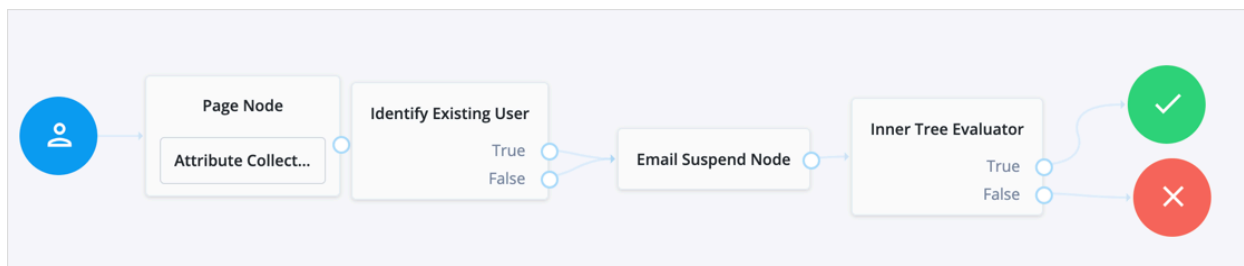
Use this node in a forgotten password flow to fetch a username to email to the user. To display the username on the screen, use the [Display Username node](#) instead.

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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

To set the number of KBA questions, edit **Configure > Security Questions > Questions > Number** in the IDM admin UI.

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.


NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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

Properties

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

Property	Usage
Questions	<p>Create or modify custom localized questions that the user can choose from when defining security questions.</p> <p>To add a localized security question:</p> <ol style="list-style-type: none"> 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. <p>To edit an existing security question, click the edit icon , make your changes, and click Save.</p>

KBA Verification node

Presents KBA questions to the user, collects answers to those questions, and verifies the input against the user's stored answers.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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.

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.

Passthrough 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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

Before you use the node:

- Configure the connector to the third-party service.

For details, refer to [Connectors](#) in the IDM 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](#) to collect the username and password.

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

Outcomes

- Authenticated
- Missing Input
- Failed

Properties

Property	Usage
System Endpoint	Required. Name of the connector to the third-party service that performs authentication.

Property	Usage
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

Patch Object node

Patches the attributes in an existing managed object in IDM.


NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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

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	<p>Fields from the shared node state that should be ignored as part of patch.</p> <p>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.</p>
Identity Resource	<p>The type of IDM managed identity resource object that this node creates.</p> <p>It must match the identity resource type for the current flow.</p> <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p>TIP</p> <p>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.</p> <p>Identity managed object types are preceded by the  icon.</p> </div>
Identity Attribute	<p>The attribute used to identify the object to update in IDM.</p>

Platform Password node

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

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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.

Outcomes

Single outcome path.

Properties

Property	Usage
Validate Password	<p>When enabled, this node checks the user's input against IDM's password policies, and returns any policy failures as errors.</p> <p>For example, if you submitted an invalid password on registration, the response from this node would include a list of failed policies:</p> <pre>{ "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\" }"] }</pre>
Password Attribute	<p>The attribute used to store a password in the IDM object.</p>
Confirm Password	<p>Enable this option to require the user to enter the password identically in a second field.</p> <div style="border: 1px solid #00aaff; padding: 5px;"><p>NOTE</p><p>This property only appears when the node is placed within a Page node.</p></div>

Platform Username node

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

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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.
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 .

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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.

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.


NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

Outcomes

- True
- False

Properties

Property	Usage
Identity Resource	<p>The type of IDM managed identity resource object this node creates. It must match the identity resource type for the current flow.</p> <p>TIP</p> <p>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.</p> <p>Identity managed object types are preceded by the  icon.</p>

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.

The node has two possible outputs: social authentication and local authentication. Local authentication can be turned off by disabling **Include local authentication**.

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.

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	ForgeRock Identity Platform deployments only.
Password attribute	ForgeRock Identity Platform deployments only.
Identity Attribute	ForgeRock Identity Platform deployments only.
Filter Enabled Providers	<p>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.</p> <div style="border: 1px solid green; padding: 5px; margin: 10px 0;"> <p>TIP</p> <p>View the names of your configured social identity providers in AM admin UI under Realms > Realm name > Services > Social Identity Provider Service > Secondary Configurations.</p> </div> <p>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.</p>

Terms and Conditions Decision node

Verifies the user has accepted the active set of terms and conditions.

NOTE

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

You set up terms and conditions in the 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](#).

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.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

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.

Outcomes

- True
- False

Properties

Property	Usage
Elapsed Time	<p>The amount of time since the user was created, in minutes, that needs to elapse before this node is triggered.</p> <p>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.</p>
Identity Attribute	<p>The attribute used to identify the object to update in IDM.</p>

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.

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.

Outcomes

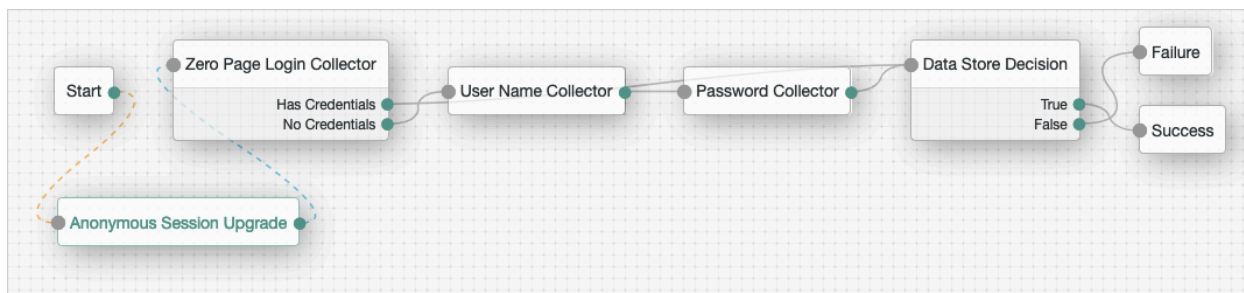
Single outcome path.

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.

Outcomes

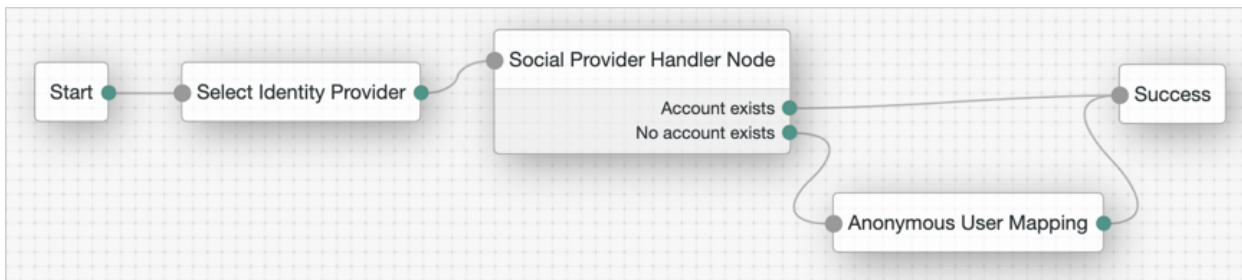
Single outcome path.

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.

Outcomes

- Choice 1
- ...
- Choice n

Properties

Property	Usage
Choices	<p>Enter two or more choice strings to display to the user.</p> <p>To remove a choice, select its Delete icon ×.</p> <p>To delete all choices, select the Clear all button in the Choices field.</p>

Property	Usage
Default Choice <i>(required)</i>	<p>Enter the value of the choice to be selected by default.</p> <div style="border: 1px solid purple; padding: 5px;"> <p>IMPORTANT</p> <p>If you do not specify a default choice, the first choice in the list becomes the default.</p> </div>
Prompt <i>(required)</i>	Enter the prompt string to display to the user when presenting the choices.
Field Display Type	<p>Specifies the format of the options presented to the user.</p> <div style="border: 1px solid blue; padding: 5px;"> <p>NOTE</p> <p>This property only appears when the node is placed within a Page node.</p> </div> <p>Possible values are:</p> <p><i>select</i> Lets the user select one or more options from a selection (default).</p> <p><i>radio</i> Lets the user select a single option from a group of radio buttons.</p>

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?
ForgeRock Identity Cloud	✓
ForgeRock Access Management (self-managed)	✓
ForgeRock 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 AM admin UI, use the [API explorer](#) endpoint `/realm-config/authentication/authenticationtrees/nodes/NodeType#_action_template`.

The following request returns the configuration properties for a [Message node](#):

```
$ curl \
--request POST \
--header "<cookie>: <token>" \
"https://openam.example.com:8443/openam/json/realm-
config/authentication/authenticationtrees/nodes/MessageNode?
_action=template"
{
  "messageYes": {},
  "message": {},
  "messageNo": {}
}
```

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

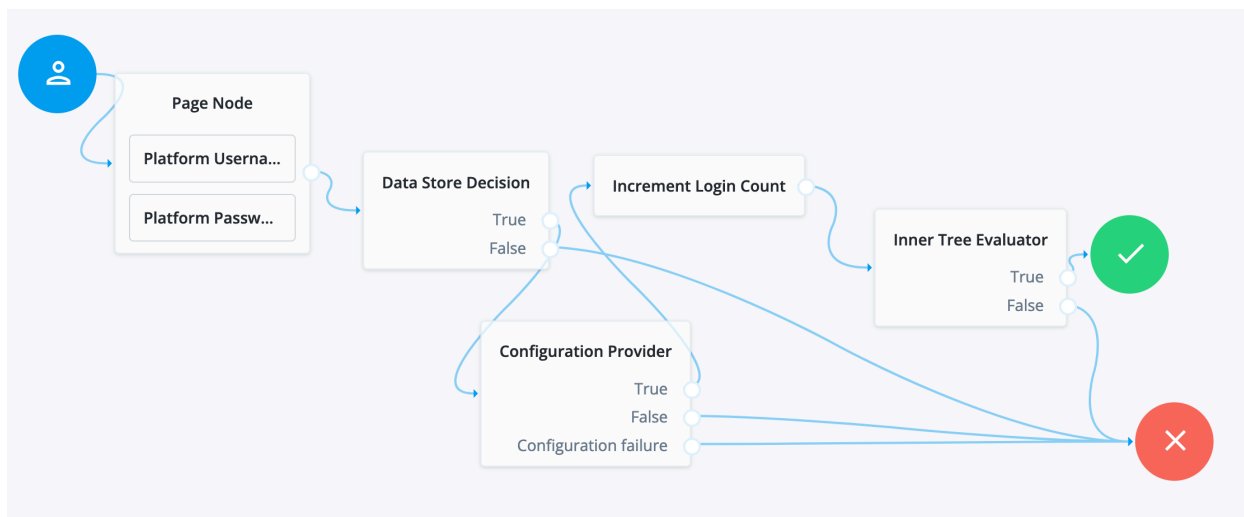
Message Node

Script Inputs

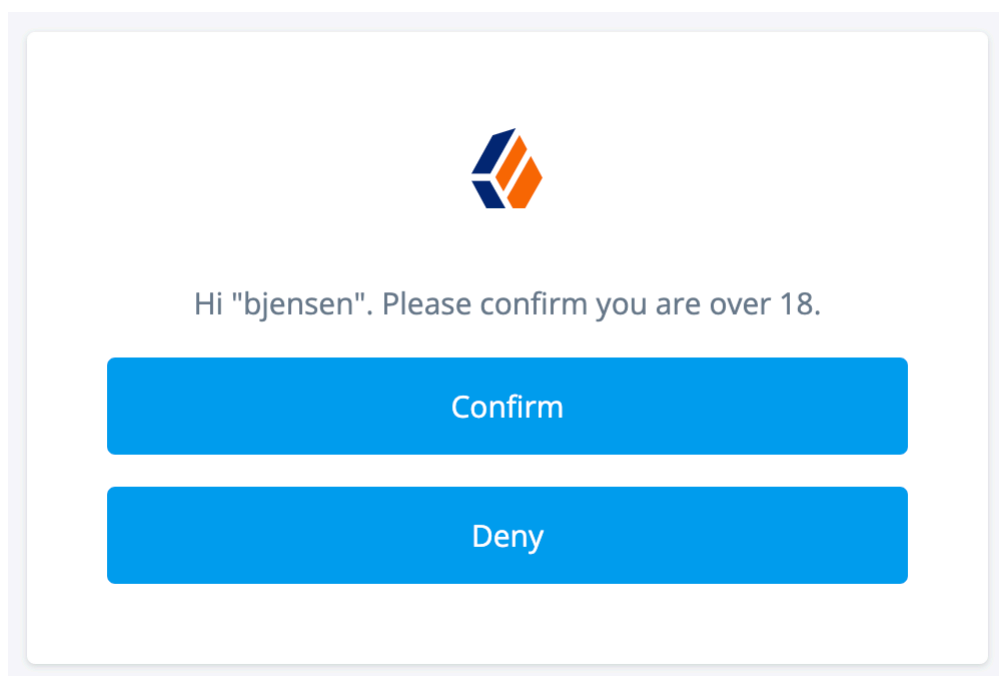
username

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

Generates and sends mail to a user, such as an address verification email, based on an email template in IDM. Authentication pauses until the user clicks a link in the email to resume the flow.

NOTE

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

This node generates the link and passes it to IDM as the `resumeURI` property of the email object. It uses the email service configured in IDM to send email. If there is no need to pause authentication and wait for a response from email, use the [Email Template node](#) instead.

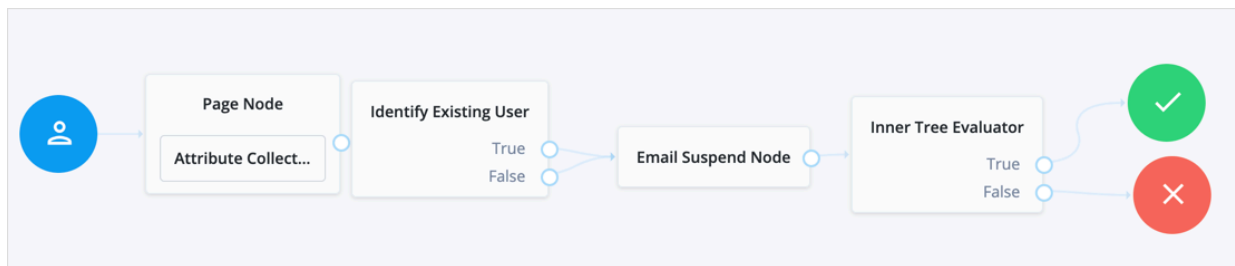
Properties

Property	Usage
Email Template Name	The name of the IDM email template. Check IDM for the names of available email templates, or to create a new template.
Email Attribute	The IDM attribute storing the address to send the email to.
Email Suspend Message	The localized message to return once AM suspends authentication. The default message is, "An email has been sent to your inbox."
Object Lookup	Determines whether to look up the object in IDM. If enabled, AM queries IDM for an existing object; otherwise, the node uses the object in the shared node state. For example, if the flow suspends user registration before creating the user object, disable this option. If registration has created the new user object while authentication was suspended, enable this option.
Identity Attribute	The attribute used to identify the object in IDM.

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 mail to the user and suspend authentication. Once authentication resumes, AM sends the user to a different flow to reset their password:



Email Template node

Generate and send an email to a user, such as a welcome email, based on an email template in IDM.

NOTE

This functionality requires that you configure AM as part of a [ForgeRock Identity Platform deployment](#).

This node uses the email service configured in IDM to send an email. If authentication should pause and wait for a response from email, use the [Email Suspend node](#) instead.

Outcomes

- Email Sent
- Email Not Sent

According to [OWASP authentication recommendations](#), the message to the user should be the same in both cases.

Properties

Property	Usage
Email Template Name	The name of the IDM email template. Check IDM for the names of available email templates, or to create a new template.
Email Attribute	The IDM attribute storing the address to send the email to.
Identity Attribute	The attribute used to identify the object in IDM.

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 [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](#).

Outcomes

Single outcome path.

Properties

Property	Usage
Failure URL (<i>required</i>)	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. For more information on upgrading a session, refer to [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";
}

```

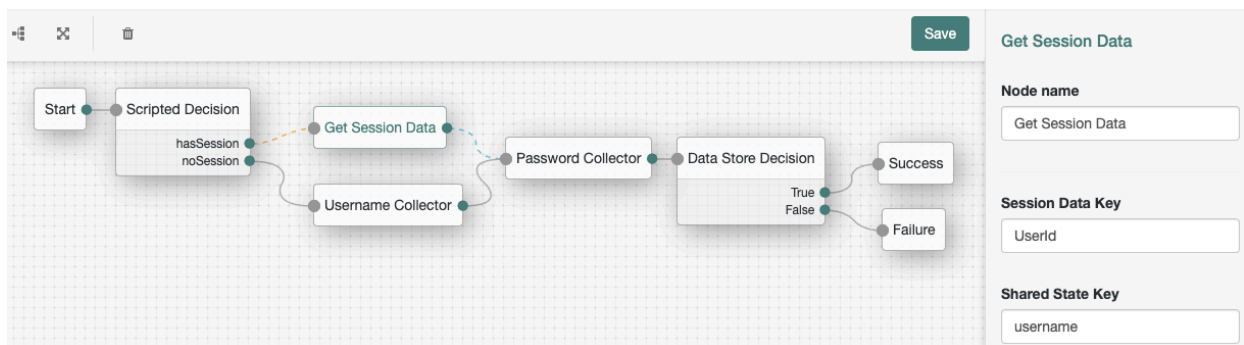
Outcomes

Single outcome path.

Properties

Property	Usage
Session Data Key <i>(required)</i>	Specify the name of a key in the user's session data used to retrieve the value.
Shared State Key <i>(required)</i>	Specify the name of a key in the <code>nodeState</code> 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

Key	Sample value
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 to 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.

For information about shared state data, refer to [Access shared state data](#).

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 (<i>required</i>)	Enter the name of the tree to evaluate.

Message node

Presents a custom, localized message to the user.

In addition to the message, you can provide localized positive and negative responses the user must select to proceed.

Outcomes

- True
- False

Properties

Property	Usage
Message	<p>Click Add. Enter the message locale in the <code>Key</code> field; for example, <code>en-gb</code>. Enter the message to display to the user in the <code>Value</code> field.</p> <p>Locales that you specify here must be <i>real</i> locales; otherwise, AM returns an <code>Invalid config</code> error.</p> <p>If the locale of the user's browser does not match any locale configured in the node, the node uses the Default Authentication Locale (set, per realm, in Authentication > Settings > General). If there is no default authentication locale, the node uses the Default Locale (set in Deployment > Servers > Server Name > General > System).</p> <p>If the message property is left blank, the text <code>Default message</code> is displayed to the user.</p> <p>To remove a message, select its delete icon (🗑).</p>

Property	Usage
Positive answer	<p>Specify a positive answer that causes evaluation to continue along the <code>True</code> outcome path.</p> <p>Click the Add button, and then enter the locale of the positive answer in the <code>Key</code> field, and the message to display to the user in the <code>Value</code> field.</p> <p>If the locale of the user's browser cannot be determined during authentication, the first message in the list is used.</p> <p>If the message property is left blank, the text <code>Yes</code> is displayed to the user.</p> <p>To remove a message, select its delete icon (🗑️).</p>
Negative answer	<p>Specify a negative answer that causes evaluation to continue along the <code>False</code> outcome path.</p> <p>Click the Add button, and then enter the locale of the negative answer in the <code>Key</code> field, and the message to display to the user in the <code>Value</code> field.</p> <p>If the locale of the user's browser cannot be determined during authentication, the first message in the list is used.</p> <p>If the message property is left blank, the text <code>No</code> is displayed to the user.</p> <p>To remove a message, select its delete icon (🗑️).</p>
Shared State Property Name	The name of the shared state variable.

Example



Do you want to join our VIP program?

Yes, please!

No, thanks!

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](#).

Outcomes

Single outcome path.

Properties

Property	Usage
Metric Key (<i>required</i>)	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.

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.

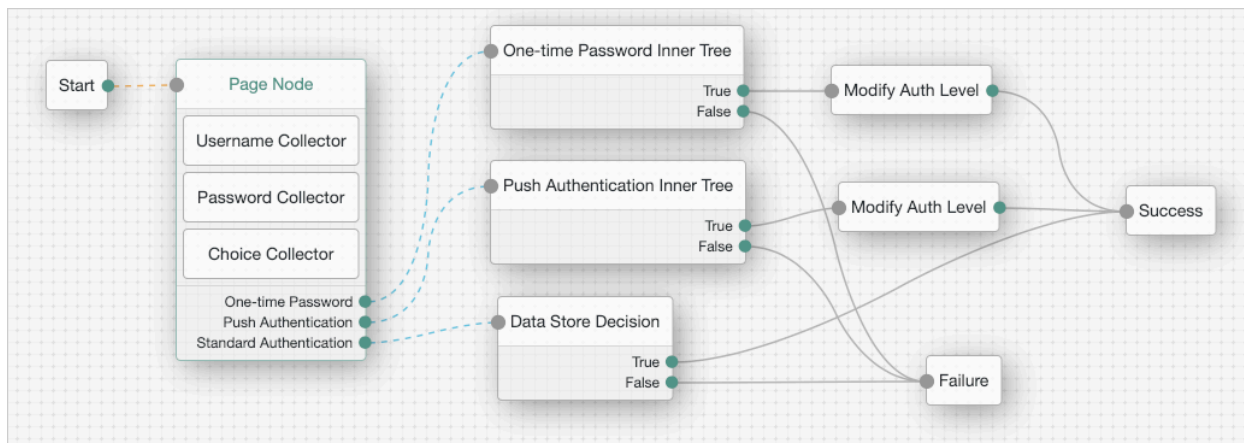
Property	Usage
Stage	An optional stage name to pass to the client to aid in rendering.

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:



User Name

Password 

Authentication type

- One-time password
- Push authentication
- Standard authentication

Log in

info@forgerock.com Copyright © 2010-2022
ForgeRock AS. All rights reserved.

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/auth
enticate?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.

Outcomes

- Done

- Exited (configurable)
- Spam (configurable)

Evaluation continues along the Done outcome path when the next request is received after the wait time has passed.

Enabling Spam detection adds a Spam outcome path to the node. Evaluation continues along the Spam outcome path if more than the specified number of requests are received during the wait time.

Enabling the user to exit without waiting adds an Exited outcome path to the node. Evaluation continues along the Exited outcome path if the user clicks the button that appears when the option is enabled. The message displayed on the exit button is configurable by using the **Exit Message** property.

Properties

Property	Usage
Seconds To Wait	Specify the number of seconds to pause authentication. Default: 8
Enable Spam Detection	Specify whether to track the number of responses received during the wait time, and continue evaluation along the Spam outcome path if the number specified in the Spam Tolerance property is exceeded. Default: Disabled
Spam Tolerance	Specify the number of responses to allow during the wait time before continuing evaluation along the Spam outcome path. This property only applies if spam detection is enabled. Default: 3

Property	Usage
Waiting Message	<p>Specifies the optional message to display to the user.</p> <p>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^[7]. The locale selected for display is based on the user's locale settings in their browser.</p> <p>Messages provided in the node override the defaults provided by AM.</p> <p>For information about customizing and translating the default messages, refer to Internationalization.</p>
Exitable	<p>Whether the user can exit the node during the wait period.</p> <p>Enabling this option adds a button with a configurable message to the page. Clicking the button causes evaluation to continue along the <code>Exited</code> outcome path.</p> <p>Default: Disabled</p>
Exit Message	<p>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, <code>Cancel</code> or <code>Lost phone? Use Recovery Code</code> . This property only applies if the Exitable property is enabled.</p> <p>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^[7]. The locale selected for display is based on the user's locale settings in their browser.</p> <p>Messages provided in the node override the defaults provided by AM.</p> <p>For information about customizing and translating the default messages, refer to Internationalization.</p>

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](#).

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`.

Outcomes

Single outcome path.

Properties

Property	Usage
Property Names (<i>required</i>)	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.

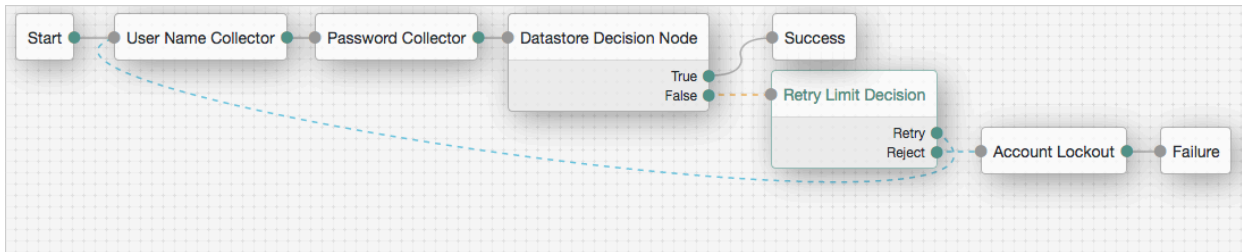
Outcomes

- Retry
- Reject

Properties

Property	Usage
Retry limit	<p>Specify the number of retries to allow.</p> <p>Default: 3</p>
Save Retry Limit to User	<p>Specify whether the number of failed login attempts persists between successful authentications. Possible values are:</p> <p>Enabled</p> <p>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.</p> <p>AM resets the count after the user authenticates successfully with a tree that contains this node.</p> <p>If AM cannot find the user's profile, authentication ends with an error.</p> <p>Disabled</p> <p>The node saves the number of failed login attempt in the <code>nodeRetryLimitKey</code> shared state property, which is discarded when the authentication session ends.</p> <p>For security reasons, ForgeRock recommends that you enable this setting.</p> <p>Default: Enabled.</p>

Example



Scripted Decision node

Runs a script during authentication.

The script defines the possible outcome paths by setting one or more values of a string variable named `outcome`. For more information on creating scripts, refer to [Manage scripts \(UI\)](#). Evaluation continues along the outcome path that matches the value of the `outcome` variable when script execution completes.

All the inputs required by the script and the outputs produced by it must be declared in the node's configuration or the script may fail. Even if the definition is `null`, it still needs to be declared. Use the wildcard `*` to include any available inputs or outputs.

For information about the API available for use in this node, refer to [Scripted decision node API](#).

Outcomes

Configurable.

Properties

Property	Usage
Script	Select the script to execute from the drop-down field.
Outcomes	Enter the possible strings that can be assigned to the <code>outcome</code> variable by the script. These strings provide the possible outcome paths.

Property	Usage
Script Inputs	<p>A list of state inputs required by the script. Defaults to <code>*</code>, which means everything currently stored in shared and transient state.</p> <div data-bbox="608 360 1401 1563" style="border: 1px solid #ccc; padding: 10px;"> <p>IMPORTANT</p> <p>Sensitive data in transient state is upgraded to <i>secure</i> state if:</p> <ul style="list-style-type: none"> * The node sends a callback to the user * A downstream node is detected that is requesting the data in the transient state as input <p>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.</p> <p>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 <i>only</i> request <code>*</code> are ignored, as this would result in putting everything that's in transient state into secure state, and retaining sensitive information longer than necessary.</p> <p>If a downstream node requires the password, it must therefore explicitly request it as state input, even if it lists the <code>*</code> wildcard as input.</p> </div>
Script Outputs	A list of state outputs produced by the script. Defaults to <code>*</code> , which means everything currently stored in state.

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](#).

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

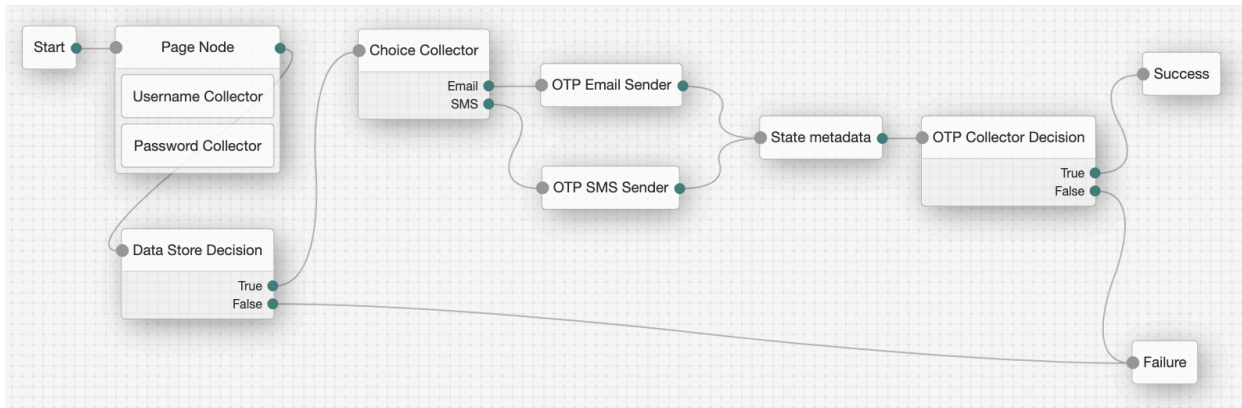
Returns selected attributes from the shared node state as metadata.

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.



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 [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](#).

Properties

Property	Usage
Success URL (<i>required</i>)	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](#).

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](#).

Outcomes

Single outcome path.

Properties

Property	Usage
Start Time Property	Specify the property name containing the time from which to calculate the elapsed time.
Metric Key <i>(required)</i>	<p>Enter the name for a new metric that stores the calculated elapsed time.</p> <p>The name that you select is used to identify the metric that exposes the data collected by this node. For example, if you enter <code>calculated.time</code>, 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 <code>am_</code> and appended with <code>_seconds</code> to become <code>am_calculated_time_seconds</code>.]</p> <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p>TIP</p> <p>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.</p> <p>For more information, refer to Audit information.</p> </div>

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, ensure that the `ref:am:reference:global-services-configuration.adoc#global-iot[IoT Service]` is configured 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](#).

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	<p>Choose the required JWT authentication method:</p> <p><i>Proof of Possession</i></p> <p>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.</p> <p><i>Client Assertion</i></p> <p>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.</p>

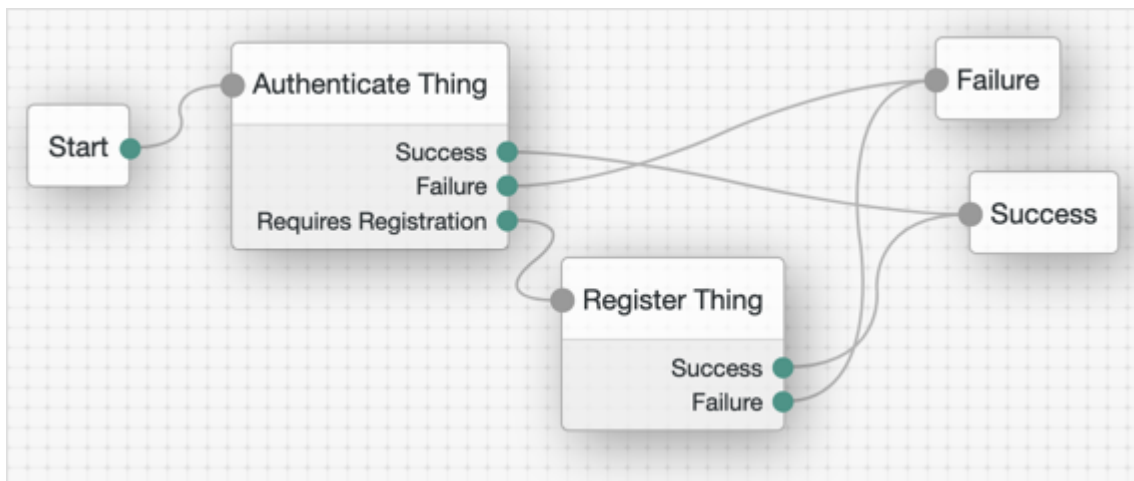
Property	Usage
Issue Restricted Token	<p>If this setting is enabled, the node adds a Proof of Possession restriction to the session token issued on successful authentication.</p> <p>Any requests accompanied by the token must be signed with the key that was used to sign the authentication JWT.</p>
Additional Audience Values	<p>Specify any additional audience values that will be permitted when verifying JWTs.</p> <p>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.</p>

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, ensure that the `ref:am:reference:global-services-configuration.adoc#global-iot[IoT Service]` is configured 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](#).

Outcomes

- Success
- Failure

Properties

Property	Usage
JWT Registration Method	<p>Choose the method to validate the JWT:</p> <p><i>Proof of Possession & Certificate</i> 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.</p> <p><i>Proof of Possession & Software Statement</i> 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.</p> <p><i>Proof of Possession</i> 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.</p> <p><i>Software Statement</i> 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.</p> <p>Default: Proof of Possession & Certificate</p>
Verify Certificate Subject	<p>If the configured JWT registration method is <code>Proof of Possession & Certificate</code>, this option verifies that the subject provided in the JWT is the same as the X.509 certificate subject CN or UID.</p> <p>Default: Enabled</p>
Create Identity	<p>Specifies whether AM will create an ID for the thing if one does not exist.</p> <p>Default: Disabled</p>

Property	Usage
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.
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.

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.

Was this helpful?  

Copyright © 2010-2024 ForgeRock, all rights reserved.