# Protect Foundations - PingAM/AIC Integration Guide

*PingOne Protect*

| Field | Value |
|---|---|
| **Version** | 1.0 |
| **Date** | 2026-04-01 |
| **Owner** | Partner Delivery Architects |
| **Intended Audience** | Technical Consultants |
| **Distribution** | Internal/Partner |

## Related Delivery Kit Assets

- **Protect Foundations - Getting Started**
- **Protect Foundations - Fundamentals**
- **Protect Foundations - Best Practices**
- **Protect Foundations - PingFederate Integration Guide**
- **Protect Foundations - DaVinci Integration Guide**
- **Protect Foundations - Delivery Roadmap Template**
- **Protect Foundations - Delivery Playbook**

# Table of Contents

# Protect Foundations - PingAM / AIC Integration Guide

This guide describes how to integrate PingOne Protect into PingOne Advanced Identity Cloud (AIC) and PingAM authentication journeys using the official PingOne Protect nodes and the PingOne Worker Service. It assumes you've completed **Protect Foundations - Getting Started** and **Fundamentals** (Protect enabled, worker app created, initial risk policy in place).

The Delivery Kit also includes links to sample flows from the PingOne Marketplace, which can be used as reference points to help you better understand the PingAM/AIC configuration while following this guide.

This guide focuses on implementation within PingAM / AIC. For overall delivery flow, follow the Protect Foundations Delivery Playbook.

---

## 1. Overview & Objectives

**Goal**
Enable AIC / PingAM journeys to:

- Call PingOne Protect during authentication or transaction journeys.
- Use device, behavioural, and contextual data to obtain risk evaluations.
- Branch journeys based on risk level, score, or recommended action to allow, challenge, or deny access.

**What this guide covers**

- Supported server versions and components.
- Setting up PingOne workers and the PingOne Worker Service in AIC/PingAM.
- Building journeys with the three PingOne Protect nodes:
    - Initialization
    - Evaluation
    - Result
- Basic validation and troubleshooting.

# 2. Prerequisites

Ensure the **Protect Foundations** - **Getting Started** and **Fundamentals** have been completed before beginning this integration.

**Servers**

- Advanced Identity Cloud (AIC) with admin access, or
- PingAM 7.5+ using the official PingOne Protect nodes (7.2–7.4 use marketplace nodes).

**PingOne**

- An environment with PingOne Protect enabled and licensed.
- At least one risk policy configured (default acceptable initially).

**A worker application of type Worker:**

- Created via Applications → Add → Application Type = Worker.
- Roles: Identity Data Admin (and additional roles as needed).
- Enabled; Environment ID, Client ID, Client Secret recorded.

**Login Widget / SDKs (for client-side data)**

- For web journeys using the Ping (ForgeRock) Login Widget or Ping SDKs with Protect:

For more detailed guidance on SDKs, see section 5.2.1 for a how-to guide.

**Access to Sample Flows**

- The Delivery Kit includes links to example flows such as Authentication Journey with Threat Detection and CIAM Passwordless Journey that integrate PingOne Protect. These can be used as reference points while following the guide to help familiarise yourself with PingOne Protect integrations in PingAM/AIC.

# 3. Components

## 3.1 PingOne Worker Application (PingOne)

Worker app in PingOne that:

- Authenticates AIC / PingAM to call PingOne Protect.
- Is referenced by the PingOne Worker Service in AIC/PingAM.

## 3.2 PingOne Worker Service (AIC / PingAM)

An AIC / PingAM **service** that stores:

- Environment ID
- Client ID
- Client Secret label (mapped to a secret store)
- Region-specific API and Auth URLs (e.g. `https://auth.pingone.com`, `https://api.pingone.com/v1`).

## 3.3 PingOne Protect Nodes

A typical AIC / PingAM journey uses three nodes:

**PingOne Protect Initialization node**

- Initialises the PingOne Protect Web SDK on the client device.
- Starts data collection for predictors that depend on device/behaviour signals.

**PingOne Protect Evaluation node**

- Sends collated data (transaction, device, behaviour) to PingOne Protect.
- Returns risk level, score, and other risk-related details.

**PingOne Protect Result node**

- Updates the risk evaluation (e.g., SUCCESS / FAILED) and/or modifies completion status.
- Provides an anchor point for feedback and further configuration.

In most implementations, all three nodes (Initialization, Evaluation, Result) should be used to ensure full risk evaluation and feedback to PingOne Protect.

# 4. Set Up PingOne Workers & Service

## 4.1 Create Worker Application in PingOne (Recap)

If not already done (see **Protect Foundations - Fundamentals**):

In PingOne admin console:

- Go to Applications → Add.
- Set Application name (e.g. `PingAM Protect Worker`).
- Application Type = Worker; save.

In the worker app properties:

- Roles tab → Grant Roles → assign at least Identity Data Admin for the correct environment.
- Overview tab → enable the application (toggle on).

Record:

- Environment ID
- Client ID
- Client Secret

## 4.2 Register the Client Secret as a Secret

**Advanced Identity Cloud**

1. Go to Tenant Settings → Global Settings → Environment Secrets & Variables.
2. Secrets tab → + Add Secret:
    - Name: e.g. `ping-protect-client-secret`.
    - Value: the Client Secret from the PingOne worker app.
3. Save, then Apply Update so the secret becomes active.

**Self-managed PingAM**

Use key aliases / secret mapping as per PingAM security docs (Create key aliases / Map and rotate secrets).

## 4.3 Configure PingOne Worker Service

**In AIC (native consoles)**

1. Go to Native Consoles → Access Management (AM).
2. In AM admin UI, navigate to Realms → Services.
3. Add or open PingOne Worker Service.
4. Under Secondary Configurations:

- **Add Secondary Configuration** → New worker configuration:
- **Name**: e.g. `Protect Worker`.
- **Client ID**: from PingOne worker app.
- **Client Secret Label Identifier**: e.g. `workerAppClientSecret`.
- **Environment ID**: from PingOne environment.
- Save.

5. Verify PingOne API Server URL and Authorization Server URL for your region (e.g. NA: `https://auth.pingone.com`, `https://api.pingone.com/v1`).
6. Save changes.

## 4.4 Map Secret Label to Secret

**AIC**

1. In AM admin UI, go to Realm → Secret Stores.
2. Select the ESV secret store.
3. Go to Mappings → + Add Mapping:
   - Secret Label: e.g. `am.services.pingone.worker.workerAppClientSecret.clientsecret`.
   - Aliases: e.g. `esv-ping-protect-client-secret` (the secret name from step 4.2).
4. Save.

**PingAM**

Use PingAM's secret mapping facility to point the label identifier to the stored client secret.

At this point, your AIC / PingAM environment can obtain tokens for the PingOne worker app and call Protect.

After configuration, verify the Worker Service can successfully obtain a token using the configured credentials before proceeding.

# 5. Build a Journey with PingOne Protect Nodes

This section focuses on implementation patterns. For conceptual understanding of Protect components, refer to the Fundamentals guide.

## 5.1 Basic Authentication Journey Pattern

A simple pattern in AIC / PingAM:

1. PingOne Protect Initialization node
2. Credentials node (username/password, or other primary auth)
3. PingOne Protect Evaluation node
4. Decision / branching based on risk
5. PingOne Protect Result node(s) to update evaluation outcome

## 5.2 Add PingOne Protect Nodes

In the AIC / PingAM Journey / Tree editor:

**Drag PingOne Protect Initialization node:**

- Place near the top of the journey to start data collection early.
- The quality and timing of Signals SDK or device data collection directly impacts the accuracy of risk evaluations. Poor or incomplete implementation can lead to missing or low-quality signals.
- Ensure it references the appropriate PingOne Worker Service configuration (environment, worker name).

**After primary authentication:**

Add PingOne Protect Evaluation node.

Configure:

- Worker configuration (same as Initialization).
- Optional Policy ID if you want to use a specific Protect risk policy; leave blank to use default.

The Evaluation node must execute before any risk-based branching or step-up logic is applied.

**Add PingOne Protect Result node(s) at the end of each branch:**

Configure node to:

- Update the risk evaluation completion status (e.g. SUCCESS, FAILED).

- Optionally override context or store results in session attributes.

The Result node is required to ensure risk evaluations are completed with a final status.

A PingOne Protect Result node must be present on all journey exit paths (success, failure, denial, etc.). Missing Result nodes will result in incomplete evaluation data and reduced model effectiveness.

Both the Evaluation and Result nodes must be implemented to ensure accurate risk evaluation and continuous model learning.

Ensure Signals SDK or device profiling is implemented early in the user interaction and allowed sufficient time to collect data before the Evaluation node executes.

### 5.2.1 Implementing Signals (Protect) SDK

Use this when device and behavioural data from the Signals (Protect) SDK must be included in PingOne Advanced Identity Cloud / PingAM journeys via the PingOne Protect nodes and PingOne Worker Service.

**1. Confirm platform and SDK compatibility**

Verify the PingAM / AIC version supports the PingOne Protect nodes:

- Native nodes in 7.5+
- Earlier versions use marketplace nodes

**Confirm access to the Signals (Protect) SDK documentation for:**

- Web (JavaScript)
- Android
- iOS

Ensure the Login Widget or SDK version in use supports PingOne Protect integration where applicable

**2. Configure PingOne worker and Worker Service**

PingOne worker application (if not already configured)

In PingOne:

- Applications → Add → Application Type = Worker
- Assign required roles (minimum: Identity Data Admin)
- Enable the application

Record:

- Environment ID
- Client ID

- Client Secret

### Store the client secret (AIC)

In AIC (Tenant Settings → Global Settings → Environment Secrets & Variables)

- Add a secret (for example esv-ping-protect-client-secret) with the worker client secret

#### Configure PingOne Worker Service

In AIC / PingAM:

- Realm → Services → PingOne Worker Service

Add a Secondary Configuration:

- Name (for example Protect Worker)
- Client ID
- Client Secret Label Identifier (for example workerAppClientSecret)
- Environment ID
- Region-specific Auth and API URLs - e.g. https://auth.pingone.com, https://api.pingone.com/v1

#### Map the secret label to the secret

Realm → Secret Stores → ESV store → Mappings

Map:

- Secret Label → am.services.pingone.worker.workerAppClientSecret.clientsecret (or equivalent)
- Alias → esv-ping-protect-client-secret

At this point, the Worker Service can obtain tokens and call PingOne Protect.

### 3. Add PingOne Protect nodes to the journey

Use the standard three-node pattern.

#### PingOne Protect Initialization node

- Add near the start of the journey (before primary authentication)
- Select the configured PingOne Worker Service
- This initialises device and behavioural data collection

#### Primary authentication nodes

For example:

- Username / Password
- Social login
- Other primary authentication steps

**PingOne Protect Evaluation node**

- Add after primary authentication input but before MFA / step-up

Configure:

- Worker configuration (same as Initialization)
- Optional Risk Policy ID (leave blank to use default)

**PingOne Protect Result node(s)**

- Add to all journey exits (success, failure, deny, etc.)
- Configure to:
    - Update evaluation completion status (SUCCESS, FAILED, etc.)

### 4. Web journeys – Login Widget / Signals SDK integration

Use this for browser-based journeys.

**Initialise the Signals (Protect) SDK**

Example (JavaScript):

```
import Widget, { protect } from '@forgerock/login-widget';

await protect.start({
  envId: 'YOUR_ENV_ID',
  behavioralDataCollection: true
});
```

- Initialise at application start or when the Protect initialization callback is received
- Ensure SDK execution begins early so sufficient data is collected

**Retrieve device data (if required)**

```
const deviceData = await protect.getData();
```

- In most cases, the Login Widget / SDK automatically passes device data to the Protect nodes when correctly configured

**Ensure correct journey invocation**

- The application must call the journey that includes:
    - Initialization
    - Evaluation
    - Result nodes

- Do not invoke legacy journeys that do not include Protect nodes

### 5. Native mobile apps – Signals (Protect) SDK

Use this for Android / iOS applications.

## Add the Signals (Protect) SDK

Use the supported SDK for:

- Android
- iOS

### Initialise early and collect data

- Initialise at application start or first relevant screen
- Allow time for signal collection before invoking the journey

### Send SDK payload to the journey

Include:

- SDK payload (per SDK documentation)
- Relevant context (user ID, IP, transaction metadata)

### Map SDK data into the journey

Ensure:

- Initialization node is present
- Any additional attributes (e.g. external device IDs) are mapped into the evaluation via node configuration

### Validate end-to-end behaviour

Execute test journeys covering:

- Successful authentication
- Failed authentication
- Elevated-risk scenarios (where possible)

### In PingOne

- Verify risk evaluations are created
- Confirm device-dependent predictors are evaluated once sufficient traffic is present

### In PingAM / AIC

Confirm:

- Initialization node executes early
- Evaluation node returns risk results
- Result nodes execute on all exit paths

### Validate branching behaviour

- LOW → normal path
- MEDIUM → step-up
- HIGH / BOT_MITIGATION → deny or mitigation path

- If device-related predictors consistently show “not evaluated”, re-check SDK initialisation timing, journey wiring, and node placement

**7. Common Pitfalls**

**Worker Service or secret mapping misconfigured**

- Symptoms: Protect nodes fail or no evaluations are created
- Fix: Re-check Worker Service configuration and secret mapping

**Initialization node missing or placed too late**

- Symptoms: little or no device data; predictors show “not evaluated”
- Fix: place Initialization node early in the journey

**Widget / SDK not aligned to the correct journey**

- Symptoms: authentication succeeds but Protect is never invoked
- Fix: ensure the application calls the journey containing Protect nodes

**Mobile SDK integrated but not mapped**

- Symptoms: SDK present but no impact on risk evaluations
- Fix: ensure SDK payload or device identifiers are passed and mapped into the journey

**Region or environment mismatch**

- Symptoms: failures when calling PingOne Protect or no evaluations created
- Fix: verify Environment ID and region-specific endpoints match the target PingOne environment

## 5.3 Journey Branching on Risk

Use outcomes or Decision nodes connected from the Evaluation node to branch based on risk:

**Low risk**:

- Proceed to success path.
- Mark evaluation as SUCCESS via Protect Result node.

**Medium risk**:

- Route to step-up authentication (e.g., MFA).
- On success/failure, send updated status via Result node.

**High risk or BOT_MITIGATION**

- Route to deny, CAPTCHA, or strong mitigation flow.
- For bot-related outcomes, prioritise CAPTCHA or blocking.
- For other high-risk scenarios, consider step-up or denial based on policy.

If nodes expose attributes such as `level`, `score`, or `recommendedAction`, you can implement more granular branching (e.g., treat TEMP_EMAIL_MITIGATION differently from AITM_MITIGATION).

Start with simple branching on risk level before introducing more complex logic based on score or predictor details.

In addition to risk level, use recommendedAction and predictor context to refine decisions (e.g. treat BOT_MITIGATION differently from VPN or transaction-related risk).

For recommended response patterns, see the **Protect Foundations - Best Practices.**

### 5.4 Example Journey Flow (AIC)

From the Integration doc:

1. **PingOne Protect Initialize**
2. **Username / Password**
3. **PingOne Protect Evaluation**
4. **Decision**:
    - LOW → Success
    - MEDIUM → Step-up path
    - HIGH or BOT_MITIGATION → Mitigation/deny path
5. **PingOne Protect Result** node at end of each path to record final status.

## 6. Login Widget & Client-Side Data (Optional)

This section is optional and only required for advanced device and behavioural data collection. If you use the Ping (ForgeRock) Login Widget or Ping SDKs:

1. **Initialise Protect in the widget**

```
import Widget, { protect } from '@forgerock/login-widget';
await protect.start({
  envId: 'your-env-id',
  behavioralDataCollection: true
});
```

2. **Pause/Resume behavioural data** as needed:

```
protect.pauseBehavioralData();
protect.resumeBehavioralData();
```

3. **Retrieve collected data** (manual mode):

```
const data = await protect.getData();
```

4. Ensure the journey nodes / filters are configured to receive and forward this device/behavioural data to Protect; many of these steps are handled automatically by the official PingOne Protect nodes when used with the supported widget/SDK versions.

# 7. Validation & Testing

Validation should be completed before enabling enforcement decisions based on Protect.

## 7.1 Validate Connectivity

Trigger the journey and ensure:

- Nodes execute without error.
- No issues in AIC / PingAM logs regarding PingOne Worker Service or Protect calls.

## 7.2 Validate Risk Evaluations

Run several journeys (success and failure cases).

In PingOne, open Audit / Risk evaluations view and confirm:

- Risk Evaluation Created events appear for your environment.
- Data (user, IP, device, predictors, policy) is present.

## 7.3 Validate Branching

Force different risk outcomes (e.g., by using risky IPs or test conditions where available).

Confirm journey branches behave as intended:

- LOW → normal login.
- MEDIUM/HIGH → additional checks / denial.

## 7.4 Validate Feedback

Ensure Result nodes are reached on all paths:

- Risk evaluations in PingOne should show updated completion status (SUCCESS/FAILED).

# 8. Troubleshooting

Common checks:

**No risk evaluations in PingOne**

Verify:

- Worker configuration values (Env ID, Client ID, Client Secret mapping) in Worker Service.
- Login Widget/SDK correctly initialised if used.
- Network access to PingOne API/Auth endpoints.

**Node errors in journeys**

- Inspect AIC / PingAM logs for PingOne Protect node errors.
- Confirm the PingOne Worker Service is enabled and secrets mapping is correct.

**Risk evaluations missing device data**

Ensure:

- Protect Initialization node is present early in the journey.
- Login Widget / SDK scripts are properly configured and not blocked.
- JavaScript is enabled on client browsers where device profiles are required.

**Predictors frequently "not evaluated"**

Confirm:

- Required inputs (IP, device payload, external attributes) are collected and forwarded.
- Any third-party or custom attributes referenced in policies are being populated.

For more detail, refer to the **Protect Foundations - Delivery Playbook** for additional testing patterns and delivery scenarios.