Protect Foundations - DaVinci 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 - PingAM / AIC Integration Guide**
- **Protect Foundations - Delivery Roadmap Template**
- **Protect Foundations - Delivery Playbook**

Table of Contents

Protect Foundations - DaVinci Integration Guide

This guide describes how to integrate PingOne Protect into PingOne DaVinci flows using the PingOne Protect connector so you can perform risk evaluations inside DaVinci journeys and branch on the results. 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 DaVinci configuration while following this guide.

This guide focuses on implementation within DaVinci. For overall delivery flow, follow the **Protect Foundations - Delivery Playbook.**

---

1. Overview & Objectives

Goal
Enable DaVinci to:

- Collect contextual and device data (via HTTP/Forms collectors or the Signals SDK).
- Call PingOne Protect to create and update risk evaluations during a flow.
- Branch based on risk level, score, or recommended action (e.g., BOT_MITIGATION) to adapt authentication and transaction flows.

What this guide covers

- Connector components and prerequisites.
  How to configure:
  - PingOne (environment, Protect, worker app, policies).
  - DaVinci (PingOne Protect connector, collectors, subflows, branching).
- Example patterns for login, registration, and high-risk transactions.
- Validation and basic troubleshooting.

2. Prerequisites

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

PingOne

- An organization + environment with PingOne Protect added.
- PingOne Protect license (or PingOne Risk for older use cases).
- At least one risk policy configured (default is OK initially).
- A worker application with roles sufficient for Protect (typically Environment Admin + Identity Data Admin) and its Environment ID, Client ID, Client Secret recorded.

DaVinci

- Access to a DaVinci tenant mapped to the PingOne environment.
- Ability to install and configure connectors and flows.

SDK / Collector versions (if using device data)

For DaVinci web flows, supported collectors include:

- ProtectCollector / skrisk via the HTTP Connector or PingOne Forms connector
- Signals (Protect) SDK dependencies for mobile or custom apps (Android, iOS, and JavaScript), where relevant

For more detailed guidance on SDKs and collectors, see Section 5.2 for a how-to guide.

Access to Sample Flows

- The Delivery Kit includes links to example flows such as Threat Protect Subflow and PingOne for Customers Passwordless that integrate PingOne Protect. These can be used as reference points while following the guide to help familiarise yourself with PingOne Protect integrations in DaVinci.

3. Connector & Data Flow

3.1 PingOne Protect Connector (DaVinci)

The PingOne Protect connector lets DaVinci:

- **Create Risk Evaluation** – send transaction and optional SDK payload to Protect, receive level/score/details.
- **Update Risk Evaluation** – send final status (SUCCESS / FAILED, etc.) so models can learn over time.

Connector configuration requires:

- **Environment ID** – from PingOne environment properties.
- **Client ID / Client Secret** – from the PingOne worker application.

Optional inputs per evaluation include:

- **Risk Policy ID** – override default policy if needed.
- **User ID / User name / IP**.
- **Custom Attributes** – for third-party risk data or app-specific context, matching attribute names used in custom predictors.

In most flows, both Create and Update Risk Evaluation should be used to ensure Protect can both assess and learn from outcomes. Both Create and Update Risk Evaluation steps are required to ensure accurate risk assessment and continuous model learning.

3.2 Device & Context Collection

DaVinci gathers risk-relevant data via:

- **PingOne Forms connector** with device profiling enabled, or
- **HTTP Connector** using a `ProtectCollector` / `skrisk` snippet in a custom HTML template.

For mobile / native flows, you can integrate the **Signals (Protect) SDK** directly into apps and pass the payload into DaVinci via API.

Important - The quality and timing of Signals SDK or device data collection directly impacts the accuracy of risk evaluations. Poor or incomplete implementation can result in missing or low-quality signals.

Ensure Signals SDK or device profiling is implemented early in the user interaction and given sufficient time to collect data before risk evaluation is triggered.

4. Configuring PingOne (Summary)

Most work is in DaVinci, but verify in PingOne:

1. **Protect service is enabled** and visible in the environment sidebar.
2. **Worker app** is created and enabled with required roles, and:
    - Environment ID
    - Client ID
    - Client Secret
      are available for connector configuration.
3. **Risk policies** are configured:
    - Default policy as a baseline.
    - Additional policies per journey if needed (Registration, Auth, Recovery, Transactions).

If you are unsure of any of the steps above, refer to **Protect Foundations – Fundamentals.**

---

5. Configuring DaVinci

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

5.1 Add the PingOne Protect Connector

1. In DaVinci, go to Connectors.
2. Search for PingOne Protect and add the connector to your tenant.
3. Configure the connector with:
    - **Environment ID** – from PingOne environment properties.
    - **Client ID / Client Secret** – from the worker app.

Save the connector configuration.

5.2 Implementing Signals (Protect) SDK & Collectors

This section describes how to implement device and behavioural data collection for PingOne Protect in DaVinci flows. Device and behavioural data should be collected as early as possible in the user journey and mapped into the PingOne Protect connector before Create Risk Evaluation is called. This builds on the DaVinci guide's existing pattern of using Forms, HTTP/custom HTML, or SDK-based collection and mapping that payload into the connector's device input.

Ensure Signals (Protect) SDK or device profiling is implemented early in the user interaction and allowed sufficient time to collect data before the Create Risk Evaluation step executes.

Choose the appropriate method based on your architecture:

- PingOne Forms (recommended for standard web flows)
- HTTP Connector with custom HTML (skrisk / ProtectCollector)
- Signals (Protect) SDK (mobile, native, or custom apps)

5.2.1 Web Flows - PingOne Forms Connector

Use this when building standard DaVinci web journeys.

Steps

1. Add a PingOne Forms node to the flow
    - Place it at the point of user interaction (e.g. login, registration)
2. Enable device and behavioural data
    - In the Forms node, enable:
        - Device profiling
        - Behavioural data (if available)
3. Ensure Forms executes before Protect
    - The Forms node must run before the **Create Risk Evaluation** step
4. Map device data into the Protect connector
    - Open the PingOne Protect connector → Device Configurations
    - Set **Risk input from device** to the variable populated by the Forms step for device / behavioural data

5.2.2 Web Flows – HTTP Connector + Custom HTML (skrisk / ProtectCollector)

Use this when implementing custom HTML templates.

Steps

1. Add an HTTP Connector with Custom HTML
    - Place it before the PingOne Protect connector
2. Insert skrisk / ProtectCollector
    - Add the skrisk / ProtectCollector component as the first element in the HTML template
    - All other HTML must be below it
    - Configure with your Environment ID
3. Expose the device payload
    - In the HTTP connector → Output Fields List
    - Add a field (e.g. `devicePayload` or `skRiskFP`)
4. Map payload into Protect

- In the PingOne Protect connector → Device Configurations
- Set **Risk input from device** to the output field that contains the collector payload

5.2.3 Mobile / Native / Custom Apps – Signals (Protect) SDK

Use this when integrating outside of DaVinci-managed web flows.

Steps

1. Add the Signals (Protect) SDK to your application
   - Web (JavaScript)
   - Android
   - iOS
   - Use the currently supported Signals (Protect) SDK for the target platform
2. Initialise early and collect data
   - Initialise at application start or first interaction
   - Retrieve device data immediately before risk evaluation

Example (JavaScript):

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

3. Pass SDK payload into Protect

Via DaVinci flow:

- Send payload to the flow via API request
- Expose it as a variable (e.g. sdkPayload)
- In the PingOne Protect connector → **Device Configurations** → set **Risk input from device** to `sdkPayload`

Via direct Protect API:

- Include the SDK payload in the appropriate field of the risk evaluation request body, in line with the current PingOne Protect API and SDK documentation

5.2.4 Common Pitfalls

- **skrisk placement**
  - Must be first element in the HTML template
- **SDK timing**
  - Initialise early and call getData() just before evaluation
- **Incorrect mapping**
  - Ensure the variable mapped to **Risk input from device** actually contains the payload

- **“Not evaluated” predictors**
  Typically caused by:
  - Missing SDK data
  - Incorrect mapping
  - Late initialisation
- **Inconsistent Collection Method**
  - Avoid mixing Forms, custom HTML collectors, and SDK approaches within the same journey unless there is a clear design reason

5.3 Build a Basic Risk-Aware Flow

A typical login flow with DaVinci + Protect looks like:

5.3.1 Collect credentials and context

- Forms / HTTP node gathers username/password plus any extra context (e.g., transaction value, managedDevice flag).
- Device profiling runs via Forms/HTTP collector.

5.3.2 Create Risk Evaluation

- Add a **PingOne Protect connector** node with **Create Risk Evaluation** capability.
- Map inputs:
  - User Name / User ID (from collectors or ID store).
  - IP address.
  - Optional **Risk Policy ID** if not using default.
  - Optional **Custom Attributes** JSON (for custom predictors).
- The node returns:
  - `level` (LOW / MEDIUM / HIGH) `score`.
  - Predictor details and recommended action when available.

The Create Risk Evaluation step must execute before any risk-based branching or mitigation logic is applied.

All decisions (e.g. allow, step-up, deny) must be based on the result of this evaluation.

5.3.3 Branch on Risk

Use a **Decision** or **Switch** node on `result.level`, `result.score`, or `recommendedAction`:

- LOW → direct allow / no MFA.
- MEDIUM → route to MFA sub-flow.
- HIGH → route to denial, additional checks, or bot mitigation path (e.g., CAPTCHA).

5.3.4 Update Risk Evaluation

- At the end of each possible branch (SUCCESS, FAILURE, ABANDON, BOT_MITIGATION, etc.), add an **Update Risk Evaluation** step:

    - Send completion status so Protect can **learn** and refine future evaluations.
- Where available, use recommendedAction (e.g. BOT_MITIGATION) and predictor context in addition to risk level to drive more precise branching decisions.
- Do not rely solely on LOW / MEDIUM / HIGH. For recommended response patterns, see the **Best Practices guide.**

A corresponding Update Risk Evaluation step must be implemented for every evaluation.

This step must be executed on all outcome paths (e.g. success, failure, denial).

If evaluations are not completed with a final status (e.g. SUCCESS or FAILED), model learning and feedback loops will be incomplete.

Both Create and Update Risk Evaluation steps are required to ensure accurate risk evaluation and continuous model improvement.

This step is critical to ensure Protect models can learn from successful and failed outcomes.

5.4 Example Patterns

Login (Auth) Flow

Collect credentials → Create Risk Evaluation → Branch:

- LOW: issue token.
- MEDIUM: MFA → if success → Update as SUCCESS; if fail → Update as FAILED.
- HIGH: block or route to special handling, then Update with appropriate status.

Registration Flow

Collect registration details → Create Risk Evaluation (with email, IP, device data) → Branch:

- LOW: allow registration.
- MEDIUM/HIGH: reject or route to manual review; capture risk details.

High-Risk Transaction Flow

Before committing a transaction (e.g., payment):

- Call Create Risk Evaluation with transaction metadata (amount, type, managedDevice flag via custom attributes).

Branch:

- LOW/MEDIUM: allow or simple step-up.
- HIGH: require strong step-up or decline transaction.

6. Validation & Testing

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

6.1 Connectivity & Basic Evaluation

Run a test user through the DaVinci flow.

Confirm:

- The PingOne Protect connector executes without error.
- A `level` and `score` are returned in the flow context.

In PingOne, open the **Threat Protection / Protect dashboard** and confirm:

- Risk evaluations are appearing for your environment.
- Data (IP, device details, predictors) looks sane.

6.2 Risk-Based Branching

Configure simple test rules:

- For example, treat **MEDIUM / HIGH** as a separate branch.

Use test scenarios:

- Normal sign-in → expect LOW and success path.
- VPN / Tor or deliberately risky context (where possible) → aim to see MEDIUM/HIGH & verify branch behaviour.

6.3 Feedback & Learning

Confirm Update Risk Evaluation runs for:

- Successful authentications.
- Failed / cancelled flows.

In PingOne audit / dashboards:

- Verify that completion statuses are present for evaluations, enabling model learning over time.

7. Troubleshooting

Common issues and checks:

No evaluations in Protect

Verify:

- Connector Environment ID, Client ID, Client Secret are correct.
- Worker app is enabled and has sufficient roles.
- Network egress to `auth/api.pingone.*` is allowed.

Connector errors in DaVinci

Check:

- DaVinci **logs/analytics** for connector failures.
- PingOne audit logs for failed API calls.

Ensure:

- The tenant/environment mapping between DaVinci and PingOne is correct.

Device profiling/SDK not working

For web:

- Confirm Forms/HTTP connector is configured with device profiling and the right script snippets (`ProtectCollector` / skrisk).

For mobile:

- Confirm SDK dependencies and initialisation are in place (See **Protect Foundations – Best Practices**)

Predictors frequently “not evaluated”

Check:

- Required inputs (IP, SDK payload, user IDs, custom attributes) are present and mapped correctly.
- Any third-party data used in custom predictors is being provided (via custom attributes) per doc guidance.