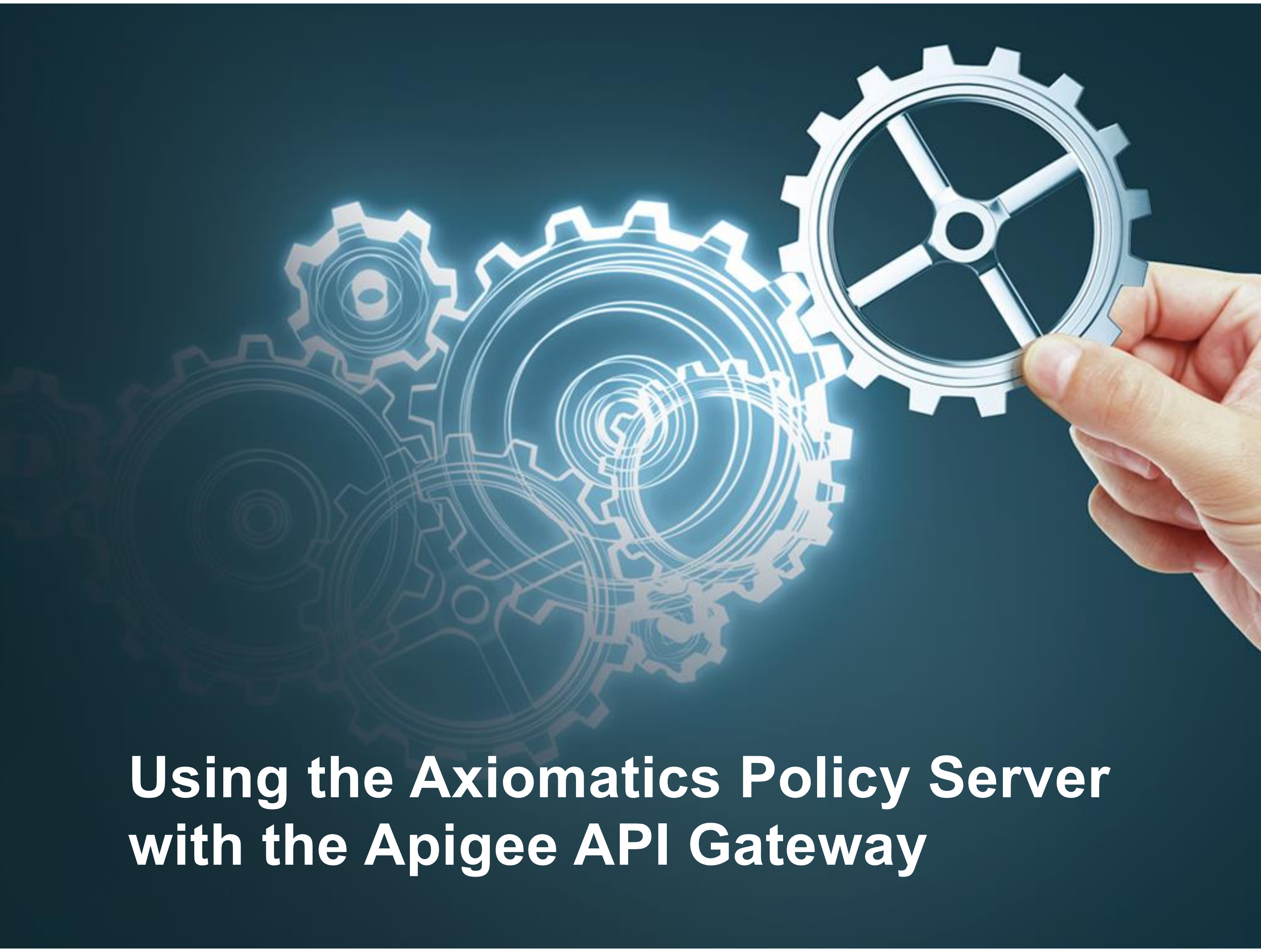
AXIOMATICS



Using the Axiomatics Policy Server with the Apigee API Gateway

Integration Guide

Table of Contents

Introduction

Add Fine-grained Authorization to the Apigee API Gateway with Axiomatics Policy Server

APIs sometimes need much finer-grained authorization decisions than are available natively in the API Gateway solution. For example, Risk, Context and Content may need to be taken into account. To achieve such granularity in authorization, it is possible to extend the API Gateway with Axiomatics Policy Server (APS) to bring Attribute Based Access Control (ABAC) to the protected APIs. Thus access to data and information is granted (or denied) based on multiple factors, which are governed by corporate policies and regulations.

By connecting the Apigee API Gateway to APS it is possible to achieve finer-grained, centrally managed authorization. The same authorization rules that are used for applications, databases, and business processes can now be applied to APIs and web services.

Lastly, fine-grained authorization can help deliver advanced scenarios such as data redaction, masking, and filtering. For instance, if an API returns a JSON payload that represents a medical record, the API gateway can process the payload and send authorization requests to the Axiomatics Policy Server to determine whether a field should be masked or redacted.

Combined Benefits with APS

The following summarizes the benefits of an integrated solution:

- Allows for fine-grained authorization policies to be expressed using the standardized XACML policy language as well as leveraging the Request/Response schema of the XACML standard.
- Any application from presentation tier to API / WS tier to business tier to data tier can leverage the powerful authorization engine from Axiomatics.
- Flexible and simple authorization policy authorirng. Two options provided:
    - Web based graphical user interface
    - ALFA Plugin for Eclipse, a developer-oriented policy editor
- Ability to use external attributes sources for runtime authorization decisions
- Tool for executing policy simulations

How it Works

The principle used in this integration is the ability of the Apigee API Gateway to make a callout to a third party service. In this case the third party service is the Axiomatics Policy Server's Policy Decision Point (PDP).

The Apigee API Gateway is configured to send fine-grained authorization requests to the Axiomatics PDP. Requests are made using the REST/JSON interface exposed by the Axiomatics PDP. The PDP then returns a response in JSON that contains the decision.

The REST interface of the PDP allows for efficient communication between the Apigee API Gateway and the Axiomatics PDP. REST is a modern architectural style that enables performance, scalability, simplicity, modifiability, visibility, portability, and reliability in distributed systems.

The JSON profile of XACML extends the Request/Response schema allowing both the Request and the Response to be encoded in JSON instead of the traditional XML encoding. This makes the Request and the Response much easier to read and also much smaller in size thus transferring less data.

Architecture

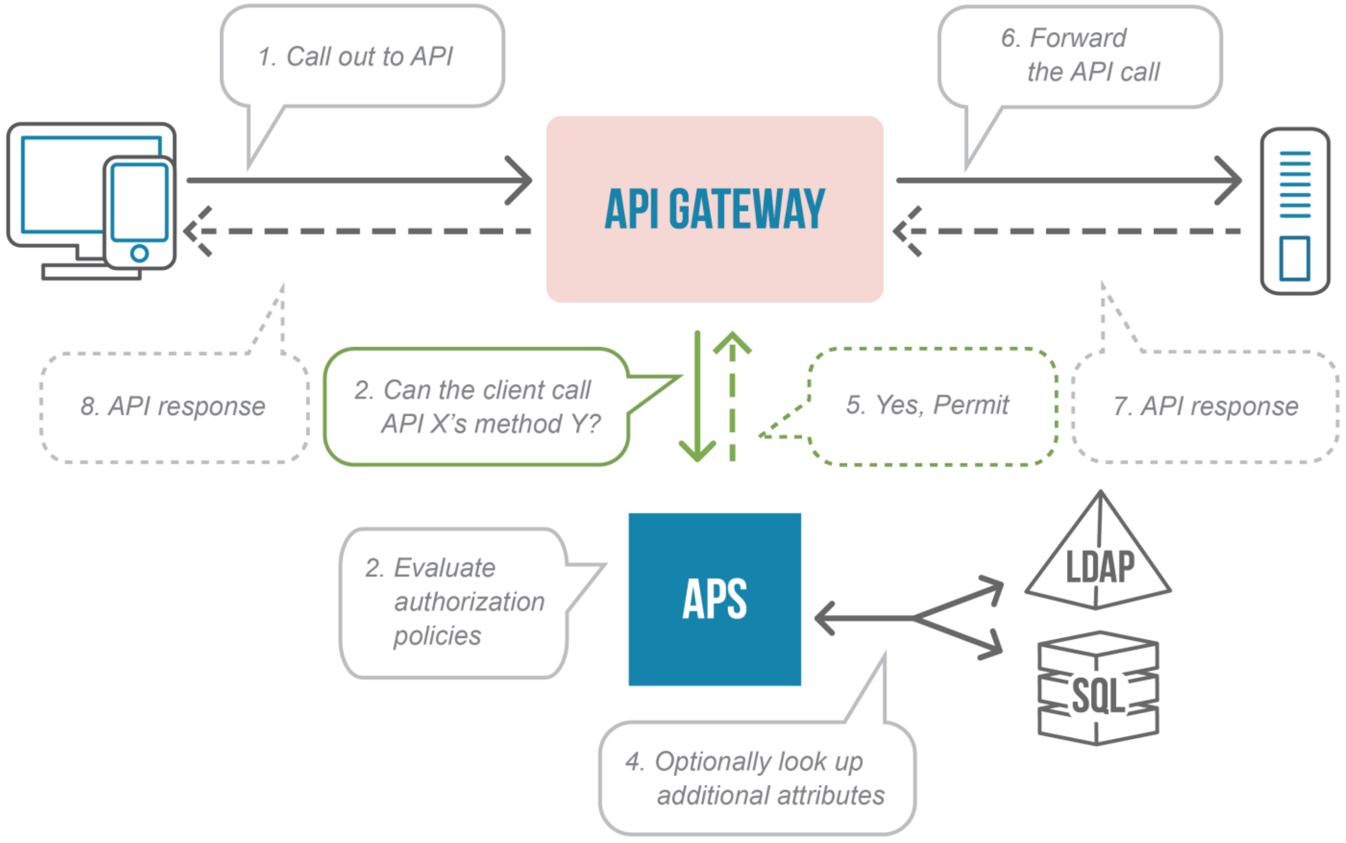

1. Call out to API
6. Forward the API call
API GATEWAY
8. API response
2. Can the client call API X's method Y?
5. Yes, Permit
7. API response
2. Evaluate authorization policies
APS
LDAP
SQL
4. Optionally look up additional attributes

Figure 1 – Call out to the Axiomatics PDP

Technical Details

Outlined below are the basic steps in the Apigee API Gateway integration policy with APS.

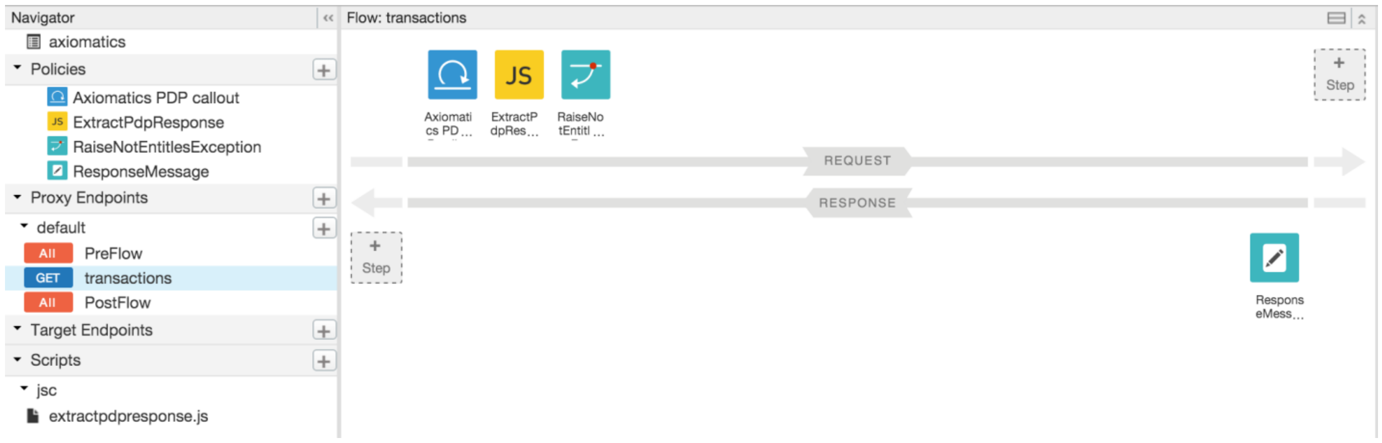


Figure 2 – An overview of the Apigee API Gateway configuration

The Callout to the PDP

In this simple configuration a callout is made to the Axiomatics PDP, the response from the PDP is captured and different actions are taken depending on the decision in the response.

The Axiomatics PDP callout configuration is available in the Apigee Management interface as XML and an example is outlined below.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ServiceCallout async="false" continueOnError="false" enabled="true"
name="CallAxiomaticsPDPService">

    <DisplayName>Axiomatics PDP callout</DisplayName>

    <Properties/>

    <Request clearPayload="true" variable="myRequest">

        <IgnoreUnresolvedVariables>false</IgnoreUnresolvedVariables>

        <Set>

            <Headers>

                <Header name="Accept">application/xacml+json</Header>

                <Header name="Authorization">Basic dXNlcjoxMjM0NTY=</Header>

            </Headers>

            <Verb>POST</Verb>
            <Payload contentType="application/xacml+json" variablePrefix="%" variableSuffix="#">

                JSON Payload

            </Payload>

        </Set>

    </Request>

    <Response>calloutResponse</Response>

    <HTTPTargetConnection>

        <Properties/>

        <URL>https://54.161.242.91:8445/asm-pdp/authorize</URL>

        <SSLInfo>

            <Enabled>true</Enabled>

            <ClientAuthEnabled>true</ClientAuthEnabled>

            <KeyStore>myKeystore</KeyStore>

            <KeyAlias>aps</KeyAlias>

            <TrustStore>myTruststore</TrustStore>

            <Ciphers/>

            <Protocols/>

        </SSLInfo>

    </HTTPTargetConnection>

</ServiceCallout>
```

The Headers Element

Two headers are added to the Request that the API Gateway constructs:

- Accept: the first is to define that the acceptable content type for the response is xacml+json and
- Authorization: the second is to add the authentication details for the PDP in Base64 coded format. As an example, if the username is `user` and the password is `123456` the Base64 encoded string would be `dXNlcjoxMjM0NTY=`

```
<Headers>

    <Header name="Accept">application/xacml+json</Header>

    <Header name="Authorization">Basic cGVwLXVzZXI6cGFzc3dvcmQ=</Header>
```

```
</Headers>
```

The Verb Element

The HTTP verb to use by the API Gateway in its communication is set to POST.

```
<Verb>POST</Verb>
```

The Payload Element

The payload segment defines that the content type of the payload is `xacml+json`. The variablePrefix and variableSuffix are defined in order for the gateway to determine where in the payload an attribute starts and ends. The actual JSON payload is described in more detail in the section The XACML Request.

```
<Payload contentType="application/xacml+json" variablePrefix="%" variableSuffix="#">

        JSON Payload

</Payload>
```

The Response Element

The response from the PDP needs to be captured and the Response element defines a name of a variable that can be accessed later to parse out the decision.

```
<Response>calloutResponse</Response>
```

The HTTPTargetConnection Element

The URL for the PDP that the API Gateway connects to is defined within the URL element of the `HTTPTargetConnection` element. Note that the port can vary depending on how the PDP is configured but the path `/asm-pdp/authorize` should be the path to use to leverage the exposed REST interface of the PDP.

If SSL is enabled the `SSLInfo` details also needs to be configured. Refer to the Apigee documentation[1] for how to configure the keystore and truststore for the gateway to use.

[1] http://apigee.com/docs/api-services/content/ssl

```
<HTTPTargetConnection>

    <Properties/>

    <URL>https://hostname:8445/asm-pdp/authorize</URL>

    <SSLInfo>

        <Enabled>true</Enabled>

        <ClientAuthEnabled>true</ClientAuthEnabled>

        <KeyStore>myKeystore</KeyStore>

        <KeyAlias>aps</KeyAlias>

        <TrustStore>myTruststore</TrustStore>

        <Ciphers/>

        <Protocols/>

    </SSLInfo>

</HTTPTargetConnection>
```

The XACML Request in our Example

In this example, the XACML request uses four gateway variables: the user identity, the request path, a query parameter and the HTTP verb. It is possible to customize this request to include additional subject, resource, action, or environment attributes. Lets break down an example API call to determine what parts that are captured in the different variables

Example API call: test.apigee.net/axiomatics/transactions?id=12&username=Alice

- `%request.queryparam.username#` – The gateway captures the end user and can leverage this variable to pass in the user as part of the request. The PDP can then use that information to further resolve attributes about the user in order to make an authorization decision. In the example API call the user is passed in as a query parameter (`username=Alice`) however this could be done in many different ways, refer to the Apigee documentation for details.
- `%message.path#` - This is the message path of the URL. In the example API URL the path would be `/axiomatics/transactions/`.
- `%request.queryparam.id#` - This is a query parameter that is captured and passed in to the gateway. In this case it's the id of the transaction that is requested and the id could then be used by the PDP to resolve further details about the transactions from underlying data sources in order to make an authorization decision.
- `%message.verb#` - The message verb is captured and is considered the action of the actual authorization request. This could be for example POST or GET.

```
{"Request":
  {"AccessSubject":
    {"Attribute":
      [
        {"AttributeId":"urn:oasis:names:tc:xacml:1.0:subject:subject-
id","Value":"%request.queryparam.username#"}
      ]
    },
  "Resource":
    {"Attribute":
      [
        {"AttributeId":"Attributes.resource.resourceType","Value":"%message.path#"},
        {"AttributeId":"urn:oasis:names:tc:xacml:1.0:resource:resource-
id","Value":"%request.queryparam.id#"}
```

```
      ]
    },
  "Action":
    {"Attribute":
      [
        {"AttributeId":"urn:oasis:names:tc:xacml:1.0:action:action-id","Value":"%message.verb#"}
      ]
    }
  }
}
```

Extracting the Decision

The gateway is configured to invoke a JavaScript that parses the response received from the PDP in order to extract the decision. The configuration simply specifies what JavaScript to execute. In this example the ResourceURL element defines that the JavaScript `jsc://extractpdpresponse.js` should be executed.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Javascript async="false" continueOnError="false" enabled="true" timeLimit="200"
name="ExtractPdpResponse">
    <DisplayName>ExtractPdpResponse</DisplayName>
    <Properties/>
    <ResourceURL>jsc://extractpdpresponse.js</ResourceURL>
</Javascript>
```

The JavaScript is added in the Script section of the Apigee management interface. A response variable is created that captures the response from the PDP (explained in section The Callout to the PDP). The response is then parsed into JSON and the Decision is captured. Example JavaScript:

```
var pdpResponse = context.getVariable("calloutResponse");
var pdpResponseJson = JSON.parse(pdpResponse);
context.setVariable("nextActionStatus",pdpResponseJson.Response.Decision);
```

Handling the Decision

There are many different actions that can be invoked depending on the Decision (Permit/Deny). In our example we configure two simple steps, one to handle a Deny decision and one to handle the Permit decision. It should be noted that these are used for illustration purposes only, in normal cases the gateway would take other types of actions such as providing the data requested when the decision is Permit for example.

A RaiseFault step called RaiseNotEntitlesException is used that displayes a message about access being denied if the decision is Deny and returns a 403 status code (HTTP status code for forbidden).

In the case of a Permit decision an AssignMessage step is invoked that displays a message that access is granted.

About Axiomatics

Axiomatics provides externalized authorization through attribute and policy-based access control for databases, applications and APIs. Our solutions are ideal for enterprises and government agencies that must securely share information (often across country borders) while complying with complex and ever-evolving regulations. Axiomatics is a leader in dynamic access control through its suite of industry leading products – the Axiomatics Policy Server and the Axiomatics Data Access Filter.

At the core of our technology lies the Externalized Access Control Mark-up Language (XACML) – we are editors for the standard and actively contribute to the development and promotion of it. We've brought together some of the brightest minds in this field to lead our Research & Development teams and ensure our solutions remain at the forefront of dynamic authorization.

Axiomatics helps our global customers within healthcare, finance, manufacturing, insurance, banking, retail, pharmaceutical, banking, retail, pharmaceutical and government agencies manage new information security challenges in cloud computing, big data and bring your own device (BYOD) trends and evolving regulatory demands.

For more information:

Feel free to contact us, follow us on Twitter or check out our website.

By email: webinfo@axiomatics.com

Twitter: http://twitter.com/axiomatics

http://www.axiomatics.com/