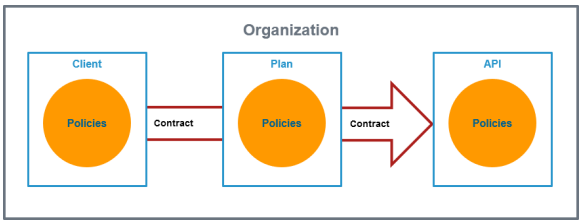
Policies

What is a Policy?

A policy is a rule or a set of rules **Scheer PAS** *API Management* uses to manage your APIs. Policies represent the unit of work done at runtime, when API Management applies the policies to all API requests.



Policies can be configured at three different levels: On an API , on a client or on a plan.

Policies are applied through a policy chain: when a request to an API is made, API Management creates a chain of policies to be applied to that request. Policy chains define a specific sequence order in which the defined policies are applied to API requests.

Expert Advice

We recommend the following best practices regarding policies:

- Give a thought or two on where to add your policy, because policies can be added to clients, plans and APIs, which has impact on the policy chain.
  - On **API level**, you will typically use modification policies, such as **URL Rewriting** or **API Key**.
  - On **plan level**, you will typically use limiting policies, such as **Rate Limiting**. This way, each plan will allow for a different amount of requests.
  - On **client level**, you will typically apply authentication and authorization policies, such as **BASIC Authentication** or **Authorization**, or other security policies.
- Testing APIs or verifying concepts with policies is much simpler with public APIs.

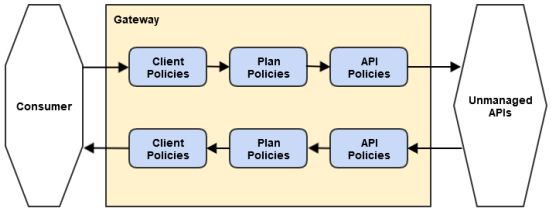
Policy Chain

A policy chain is an ordered sequence of policies that are applied when a request is made for an API through the gateway. The policies are processed in the following order:

1. **Client**
2. **Plan**
3. **API**

When a request for an API is received by the gateway, the policy chain is applied to the request in the order listed above. If none of the policies fail, the gateway will proxy the request to the backend API implementation. Once a response is received, the policy chain is then applied in reverse order to that response. This allows each policy to be applied twice, once to the inbound request and again to the outbound response. By applying the policy chain twice, API Management allows policy implementations two opportunities to provide management functionality during the lifecycle. The figure illustrates this two-way approach to applying policies.

Figure: Two-Way Policy Chain



On this Page:

Related Pages:

- Attaching Policies
- Policy Configuration
- APIs
- Clients
- Plans
- Getting Started

> From PAS 24.1 the OpenAPI definition is adapted when the API is published. Adding or removing policies enriches now the OpenAPI definition. This also applies to the whole policy chain, even if the definition editor in the API details will only show API-related policy code. Refer to API Settings > API Definition for detailed information.

Overview of the Supplied Policies

Adding a policy will allow its specific functionality to be applied to the API invocation as part of the overall policy chain. In API Management, four categories of policies are applicable:

- **Security Policies**
- **Limiting Policies**
- **Modification Policies**
- **Other Policies**

Security Policies		
Policy Symbol	**Policy Name**	**Description**
	BASIC Authentication	Enables HTTP BASIC Authentication on an API. Use this policy to require clients to provide HTTP BASIC authentication credentials when making requests to the managed API.
	CORS	Use this policy to enable and configure Cross Origin Resource Sharing on an API. This allows to access resources outside the originating domain.
	Authorization	Allows to add a list of fine-grained authorization rules. Use this policy to control precisely who is allowed to access the API.
	Header Allow /Deny	Allows you to control which incoming requests may be forwarded to the backend service. Permission is granted by adding values for a header.
	HTTP Security	Enforces transport security when using HTTP to mitigate a range of common web vulnerabilities. Contains also a sophisticated mechanism to precisely define the types and sources of content that may be loaded, with violation reporting and the ability to restrict the availability and scope of many security-sensitive features.
	Ignored Resources	Enables the user to shield some API's resources from being accessed, without blocking access to all the API's resources. Requests made to API resources designated as **ignored** result in an HTTP 404 (not found) error code. This policy allows fine-grained control over which of an API's resources are accessible.
	IP Allowlist	This policy allows access to an API's resource based on the IP address of the client. The user must specify the IP address ranges to be included from being able to access the API. Any addresses that are not explicitly allowed are not able to access the API. It is possible to use wildcard characters to specify the allowed IP addresses. It is also possible to define the return error code sent in the response to the client in case a request fails due to the violation of this policy.
	IP Blocklist	This policy blocks access to an API's resource based on the IP address of the client. The user must specify the IP address ranges to be excluded from being able to access the API. Any addresses that are not explicitly excluded are able to access the API. It is possible to use wildcard characters to specify the IP addresses to be blocked. It is also possible to define the return error code sent in the response to the client in case a request fails due to the violation of this policy. An IP Blocklist policy overrides an IP Allowlist policy.
	JWT	This policy can set headers as claim values or whole access token. It's main purpose is the validation of JWT tokens for authentication.

	Keycloak OAuth	This Keycloak-specific OAuth2 policy regulates access to APIs. It enables a wide range of sophisticated auth facilities in combination with, for instance, Keycloak's federation, brokering and user management capabilities. Keycloak's token format and auth mechanism facilitate excellent performance characteristics, with users able to easily tune the setup to meet their security requirements. In general, this is one of the best approaches for achieving security without greatly impacting performance. Refer to API Security: Authentication and Authorization for more detailed information.
	SOAP Authorization	Nearly identical to the Authorization Policy with the exception that it accepts a SOAP action in the HTTP header. This policy will only accept a single SOAP action header. It will not extract the operation name from the SOAP body.
	Time Restricted Access	Manages a list of API routes that can be accessed at specific time and date. This policy allows to control **when** client and users are allowed to access your API.
Limiting Policies		
	Rate Limiting	Governs the number of times requests are made to an API within a specified time period. The requests can be filtered by user, client or API and can set the level of granularity for the time period to second, minute, hour, day, month, or year. The intended use of this policy type is for fine grained processing.
	Transfer Quota	Tracks the number of bytes transferred. Enables the user to set a transfer quota (data) in B, KB, MB or GB for upload, download or both per client, user or API in a definable period of time. The response header can also be freely defined. The intended use of this policy type is for fine grained processing.
Modification Policies		
	JSONP	Turns a standard REST endpoint into a JSONP compatible endpoint. The caller must provide a JSONP callback function name via the URL. If the API client does not send the JSONP callback function name in the URL, this policy will do nothing. This allows managed endpoints to support both standard REST **and** JSONP at the same time.
	Simple Header	Headers can be set and removed on request, response or both. The values can be literal strings, environment or system properties. Headers can be removed by simple string equality or regular expression.
	URL Rewriting	With this policy, URLs in the request URL, the request header, the response body or the response header can be changed during a request.
Other Policies		
	API Key	Passes the API Key through to the back-end service by adding it to a customizable HTTP header.
	Caching Resources	With this policy it is possible to cache requests based on their URL path, HTTP method and HTTP status code. This allows reducing overall traffic to the backend API.
	Timeout	Allows you to determine timeouts for your API. You can differentiate between a timeout for the initial connection and a timeout for the entire request.