Foundation Patterns

The Foundation Patterns category deals with the following related executive decisions: where should the API be accessible from, and do the API clients display forms and processing results to end users or do they integrate backends?

← Evolution Patterns   Responsibility Patterns →

Category Overview

The Foundation Patterns category comprises the following patterns:

Foundation

- Frontend Integration
- Backend Integration
- Public API
- Community API
- Solution Internal API
- API Description

	PATTERN: API DESCRIPTION
Problem	Which knowledge should be shared between an API provider and its clients? How should this knowledge be documented?
Solution	Create an API Description that defines request and response message structures, error/fault reporting, and other relevant parts of the technical knowledge to be shared between provider and client. Also cover dynamic or behavioral aspects including invocation sequences, pre- and postconditions, and invariants. Complement the syntactical interface description with quality management policies, semantic specifications and organizational information.

	PATTERN: BACKEND INTEGRATION
Problem	How can parts of distributed applications that have been built independently exchange data and invoke each other's operations without breaking their system-internal conceptual integrity and without introducing undesired coupling between them?
Solution	Integrate the two (or more) backends horizontally via a message-based remote API. Apply MAPs from the other categories in this pattern language, for instance Structure Patterns, Quality Patterns, and Responsibility Patterns when doing so.

	PATTERN: COMMUNITY API
Problem	How can the visibility of and the access to an API be restricted to a closed user group that does not work for a single organizational unit (for instance, legal entities such as companies, non-profit/non-government organizations, governments)?

	PATTERN: COMMUNITY API
Solution	Deploy the API and its implementation resources securely in an access-restricted location so that only the desired user group has access to it, for instance in an extranet. Share the API Description only with the restricted target audience.

	PATTERN: FRONTEND INTEGRATION
Problem	How can client-side end-user interfaces that are physically separated from business logic and data storage be populated and updated with server-side data such as computing results, result sets from searches, and entity information? How can such frontends invoke activities in a backend and upload data to it?
Solution	Integrate frontend and backend vertically to provide Frontend Integration via a message-based remote API supported by a mature remoting technology that supports a messaging protocol and one or more message exchange formats.

	PATTERN: PUBLIC API
Problem	How can an API be made available to an unlimited and/or unknown number of API clients outside the organization (for example, globally, nationally or regionally)?
Solution	Expose the API on the public Internet along with a detailed API Description that describes both functional and non-functional aspects of the API. Use API Keys or other security means to control access to it.

	PATTERN: SOLUTION-INTERNAL API
Problem	How can access to, and usage of, an API be limited to an application, for instance components in the same or another logical layer and/or physical tier? How can an application be decomposed into services?
Solution	Decompose the application into services that expose APIs. Offer those APIs only to system-internal communication partners such as other services in the backend. Apply cutting criteria such as functional requirements and domain model but also operational requirements such as scaling needs and developmental concerns such as independent changeability.

← Evolution Patterns

Responsibility Patterns →

← BACK TO HOMEPAGE

Microservice API Patterns