Unit 5: Design of Web Services (2/2)

SOAP or Big Web Services

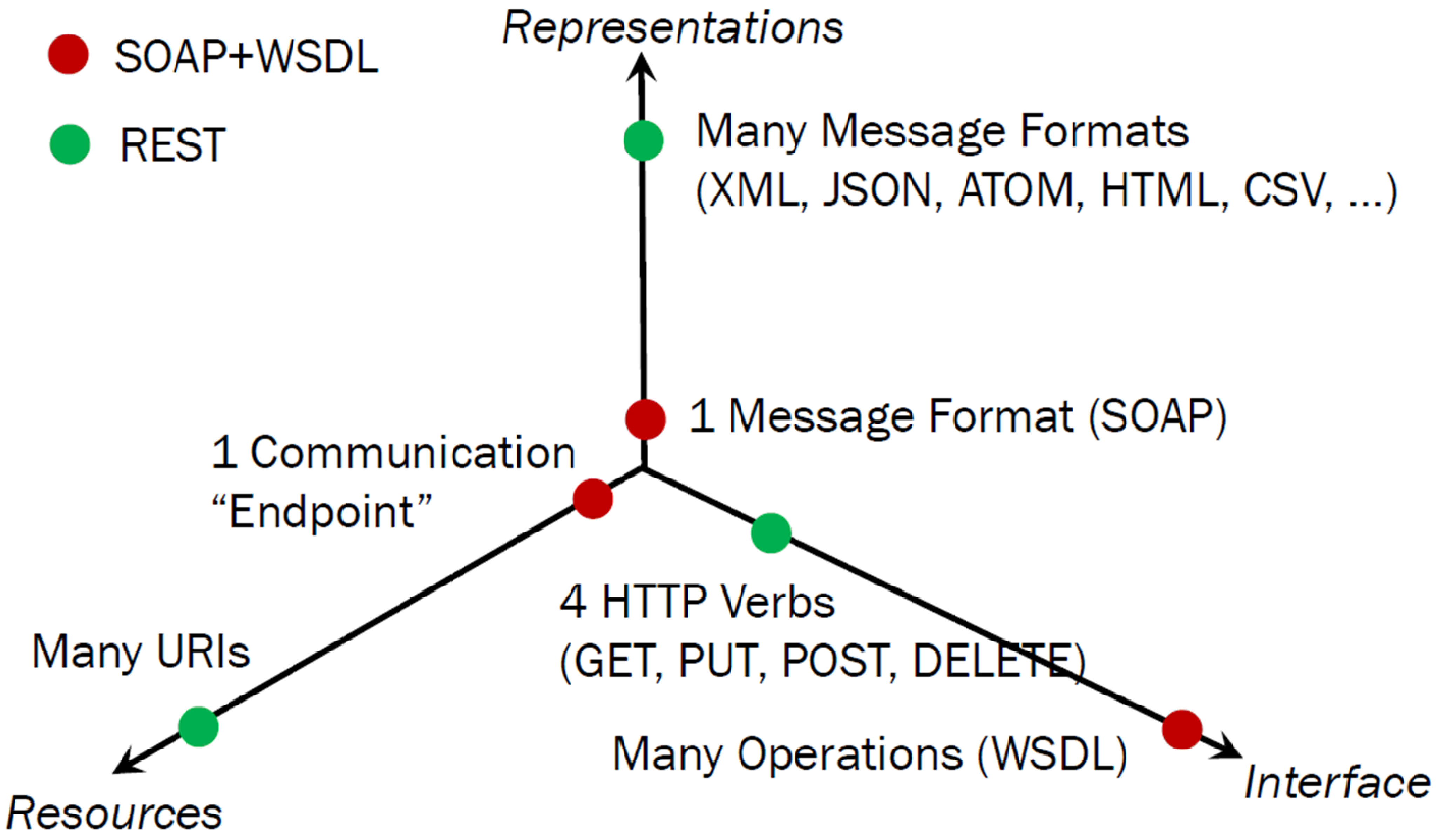
Design of Web Services

- Definition and Features of Web Services
- Types of Web Services
- RESTful Web Services
- SOAP or Big Web Services

SOAP Web Services: Intro

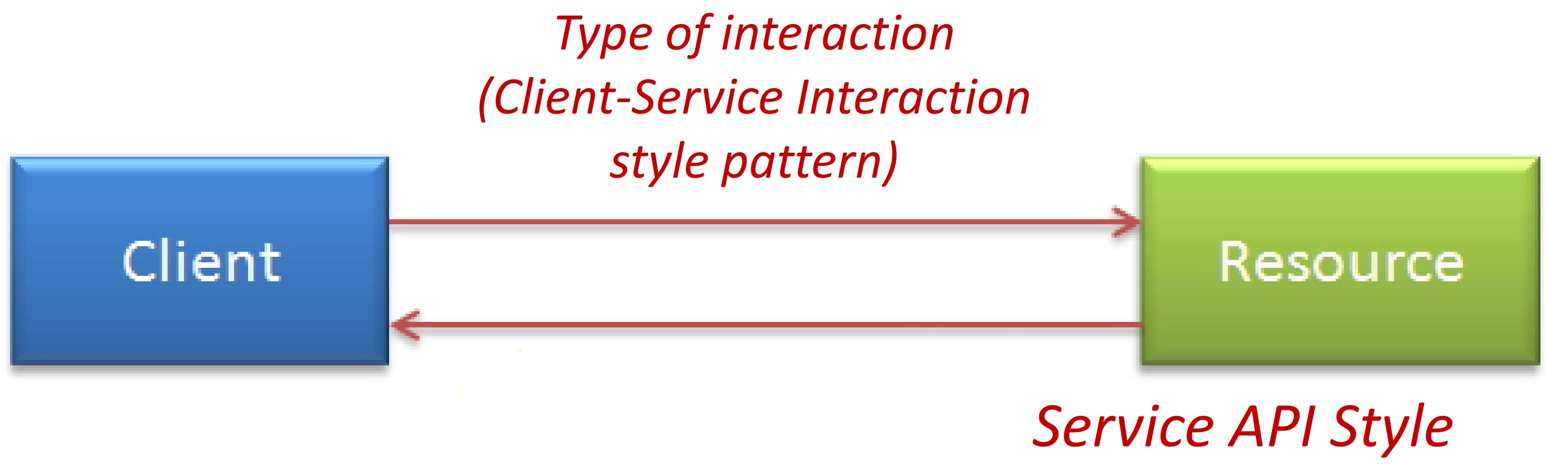
- SOAP or Big Services are web services that use XML messages that follow the SOAP standard for communication between the web service and the client.
- Web Service Description Language (WSDL) is used to define an XML file that describes the technical details of the web service, more specifically the URI, port, method names, arguments, and data types.

SOAP+WSDL vs. RESTful



Designing SOAP WS: The Roadmap

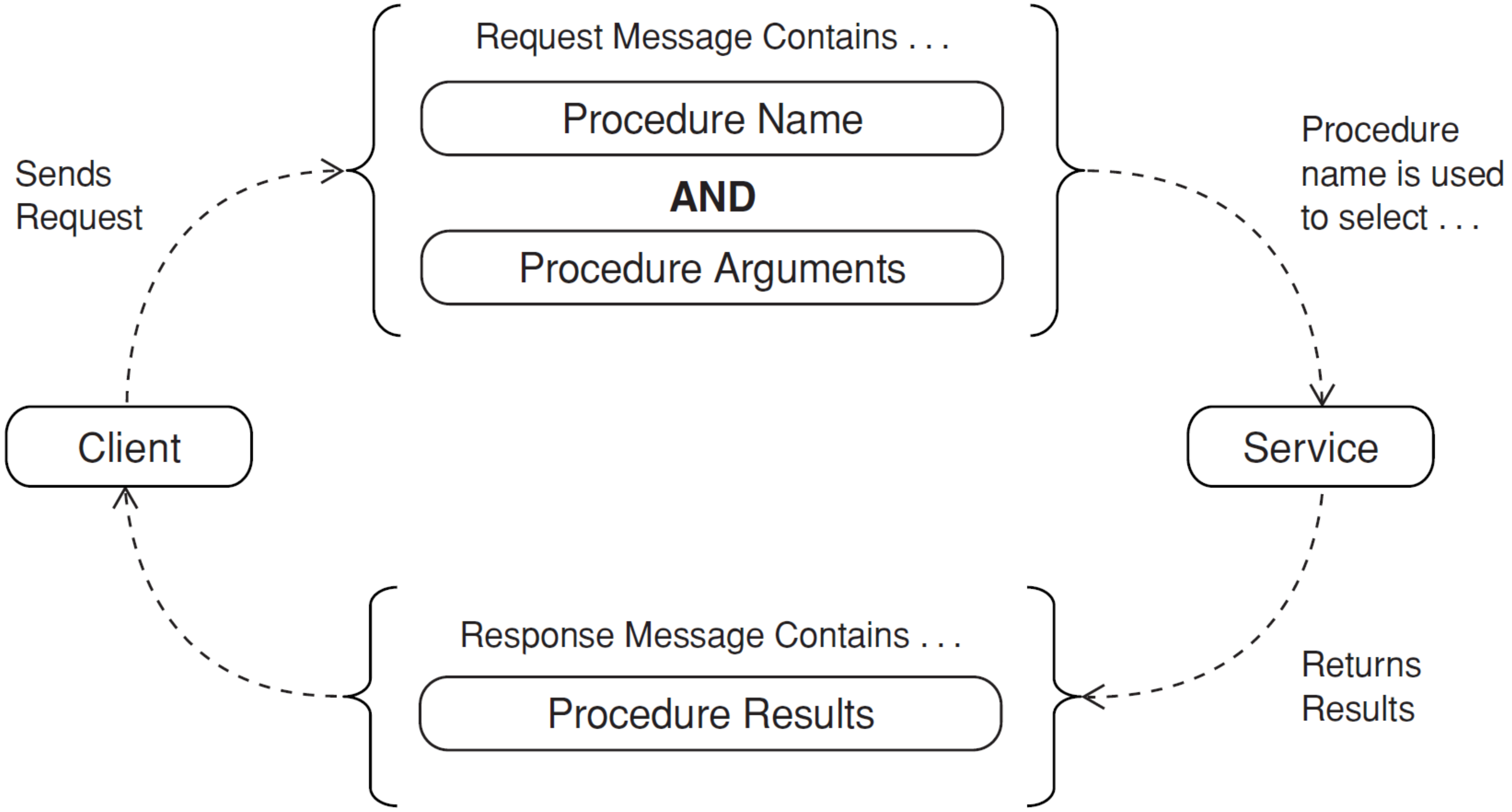
1. Select the Service API Style
2. Identify the services and their operations
3. Define the type of interaction between the client and the service
4. Apply the DTO pattern when defining the operation parameters
5. Describe the service contracts (WSDL)


Type of interaction (Client-Service Interaction style pattern)
Client
Resource
Service API Style

Selecting the Service API Style

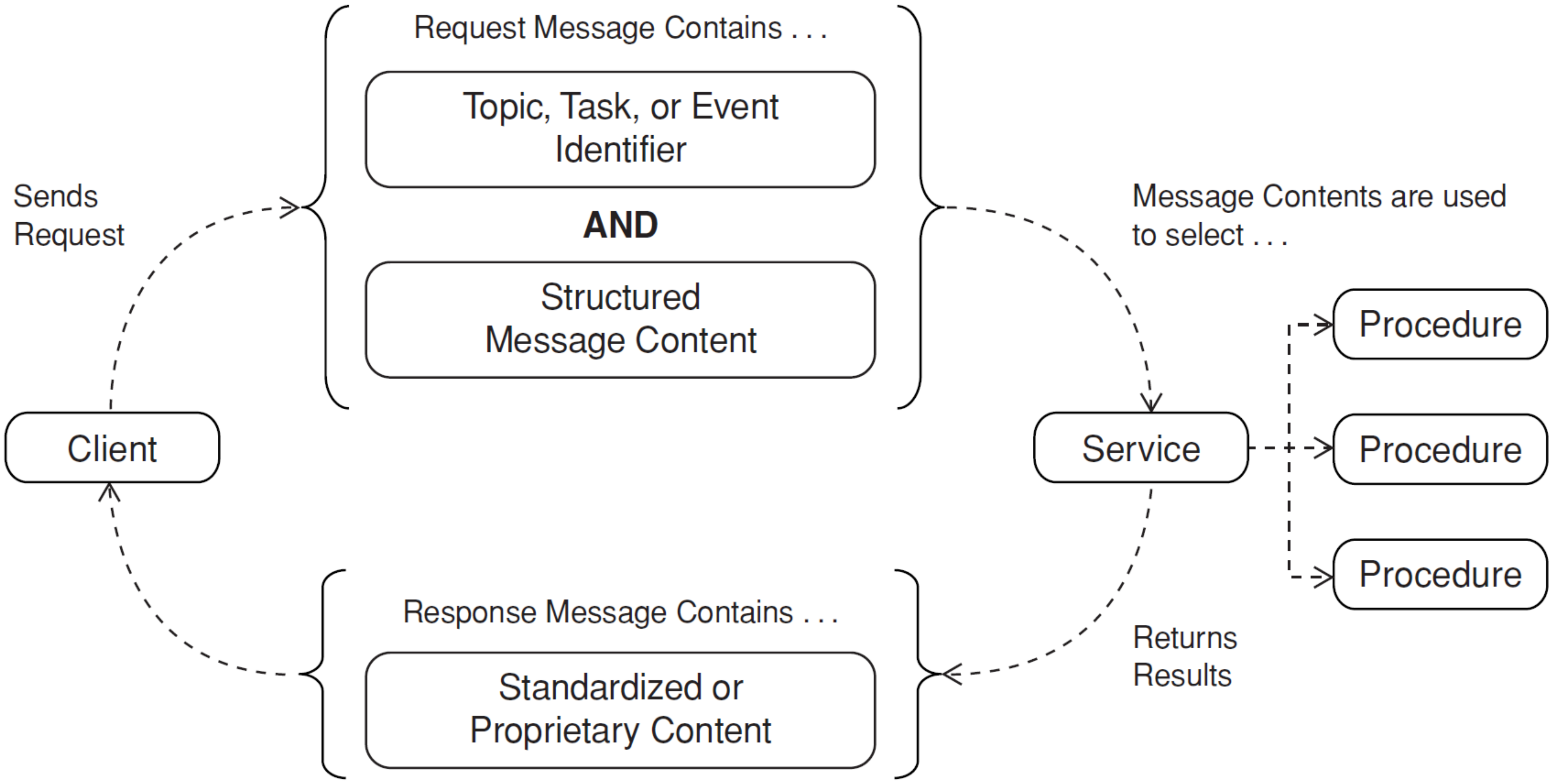
Pattern Name	Problem	Description	WS Tech.
RPC API	How can clients execute remote procedures over HTTP?	Define messages that identify the remote procedures to execute and also include a fixed set of elements that map directly into the parameters of remote procedures	• SOAP • gRPC
Message API	How can clients send commands, notifications, or other information to remote systems over HTTP while avoiding direct coupling to remote procedures?	Define messages that are not derived from the signatures of remote procedures. These messages may carry information on specific topics, tasks to execute, and events	• SOAP • AMQP
Resource API	How can a client manipulate data managed by a remote system, avoid direct coupling to remote procedures, and minimize the need for domain-specific APIs?	Assign all procedures, instances of domain data, and files a URI. Leverage HTTP as a complete application protocol to define standard service behaviors	• REST • GraphQL

RPC API


Request Message Contains . . .
Procedure Name
AND
Procedure Arguments
Sends Request
Procedure name is used to select . . .
Client
Service
Response Message Contains . . .
Procedure Results
Returns Results

* Extracted from: Robert Daigneau. Service Design Patterns. Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services. Addison Wesley, 2012

Message API


Request Message Contains . . .
Topic, Task, or Event Identifier
AND
Structured Message Content
Sends Request
Client
Message Contents are used to select . . .
Service
Procedure
Procedure
Procedure
Returns Results
Response Message Contains . . .
Standardized or Proprietary Content

* Extracted from: Robert Daigneau. Service Design Patterns. Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services. Addison Wesley, 2012

Identify services & operations

- Study the sources from which services must be derived
  - BPM, conceptual models, others
- Identify services and classify them into:
  - **Application services** represent technology and application logic (utility and wrapper services)
  - **Entity-centric services** encapsulate business entities
  - **Task-centric services** encapsulate logic specific to a task or business process
  - **Orchestration services** represent coordination of other services
- Apply service orientation design principles:
  - Definition of a service contract, granularity for reuse, low coupling, stateless services

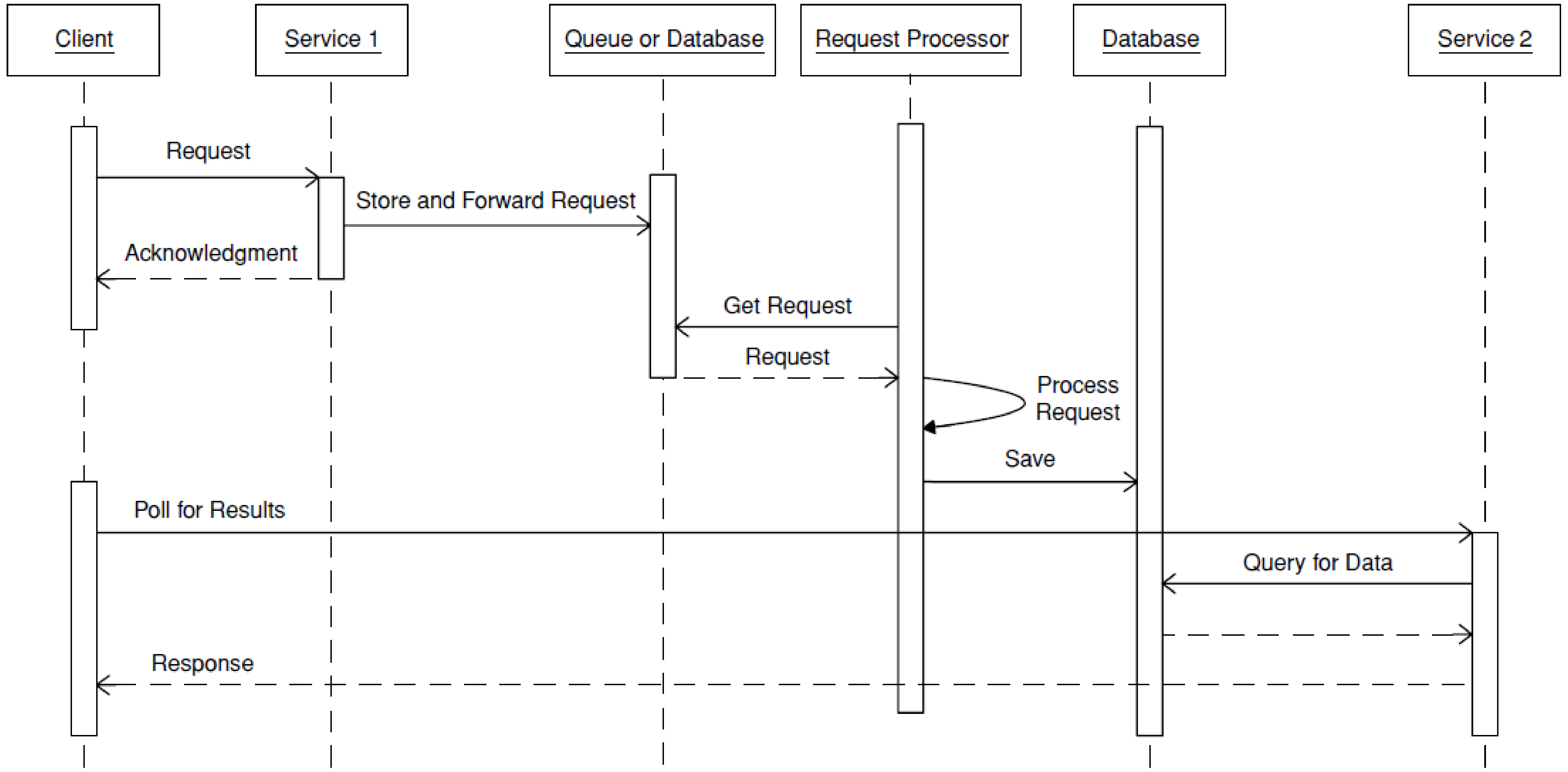
Defining the Client/Service Interaction

Pattern Name	Problem	Description
Request/ Response	What's the simplest way for a web service to process a request and provide a result?	Process requests when they're received and return results over the same client connection.
Request/ Acknowledge	How can a web service safeguard systems from spikes in request load and ensure that requests are processed even when the underlying systems are unavailable?	When a service receives a request, forward it to a background process, then return an acknowledgment containing a unique request identifier.

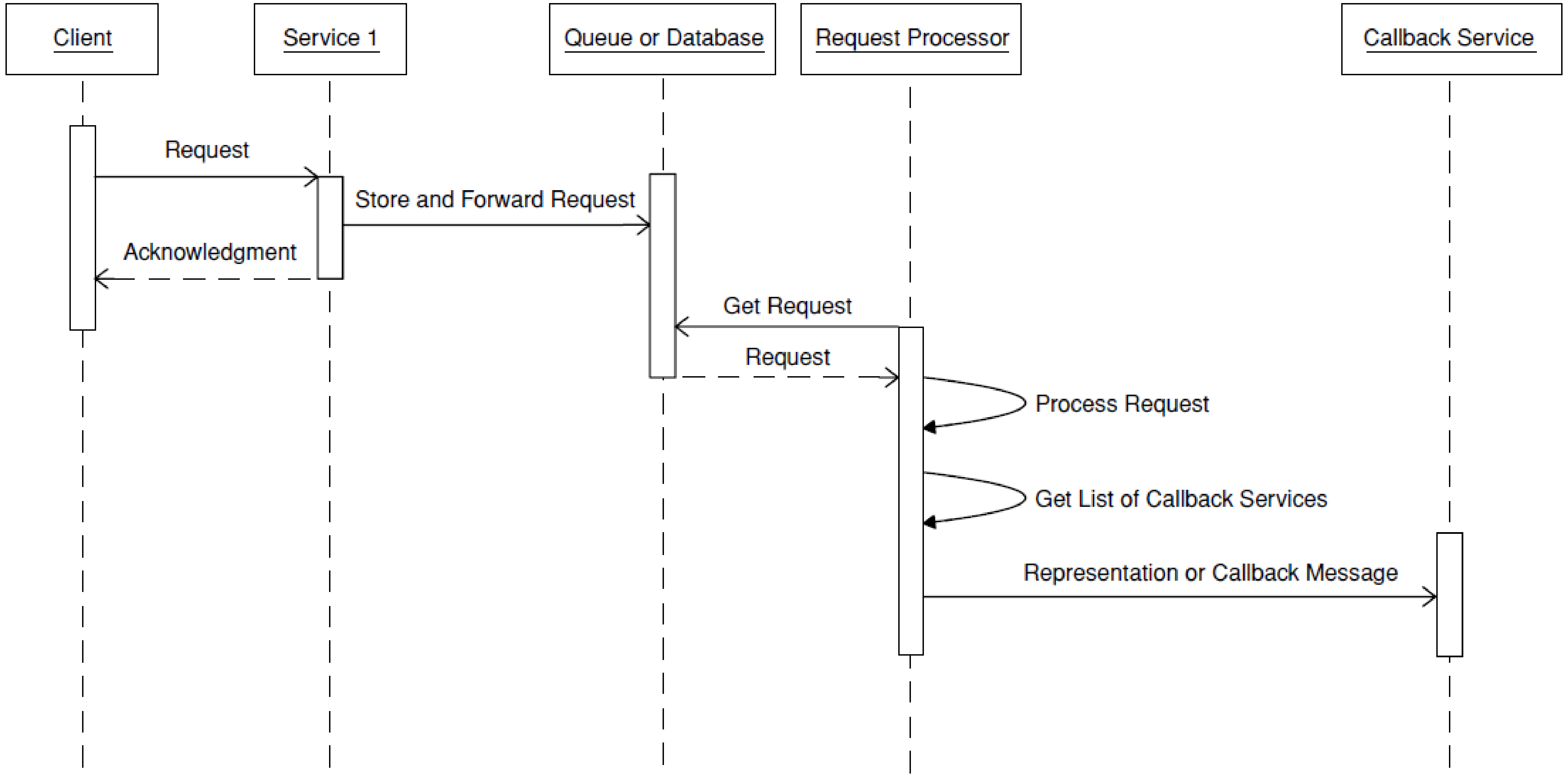
Request/Response Pattern: Issues

- High temporal coupling
  - Requests must be processed as soon as they are received
  - Workload of services. Availability and scalability issues
- Client-side blocking
  - By default, clients block and wait for responses.

Request/Acknowledge Pattern: Poll Variation



Request/Acknowledge Pattern: Callback Variation



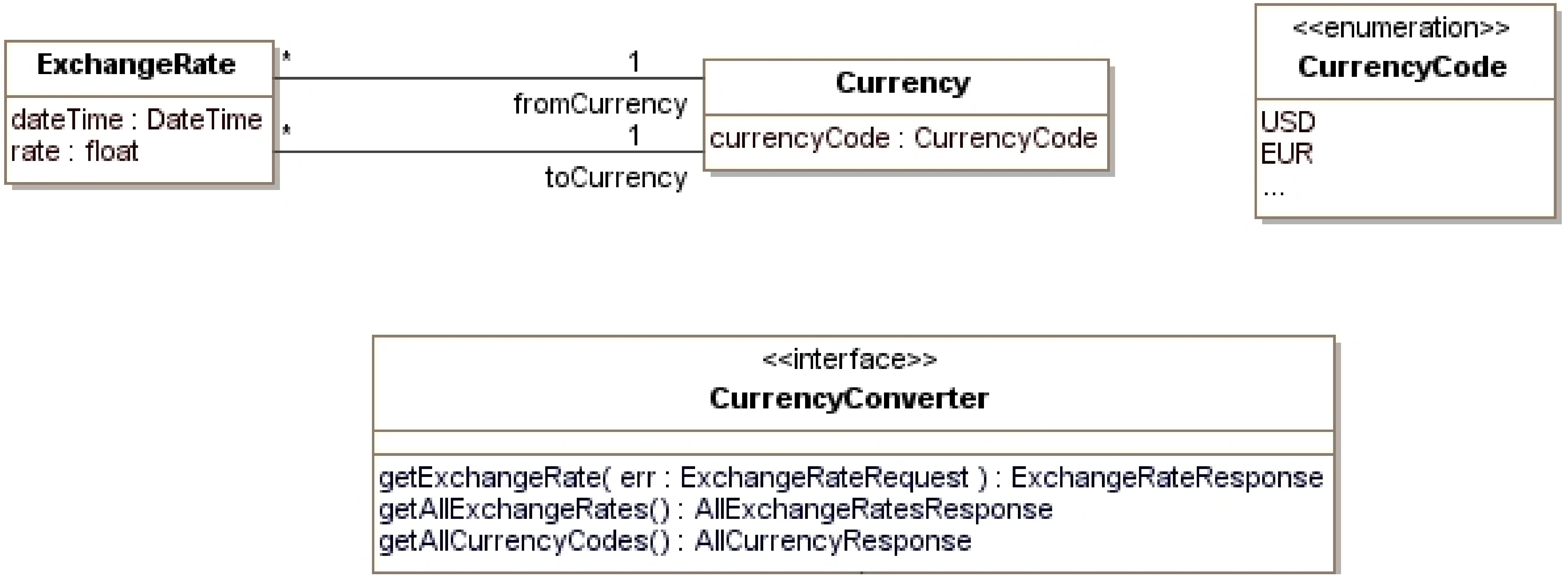
DTO Pattern

- Context
  - Systems need to communicate data to services in an efficient manner
- Problem
  - To minimize the number of calls in the system, each call has to carry more information in the parameters and in the return value
- Solution
  - We define (use) a new data group that contains all the data that has to be passed as parameter or result. The objects that are these kind of groups are called Data Transfer Objects (DTO)

Describing The Service Contract

- Formal contracts should be provided to describe services and to define terms of information exchange
- Service contracts provide definition of:
  - Service endpoint describes the point of contact for a service by stating the physical location of the service
  - Service operations
  - Input and output message supported by each operation
  - Rules and characteristics of the service and its operations
- WSDL may be used to describe service contracts

Example


ExchangeRate
dateTime : DateTime
rate : float
*
*
1
fromCurrency
1
toCurrency
Currency
currencyCode : CurrencyCode
<<enumeration>>
CurrencyCode
USD
EUR
...
<<interface>>
CurrencyConverter
getExchangeRate(err : ExchangeRateRequest) : ExchangeRateResponse
getAllExchangeRates() : AllExchangeRatesResponse
getAllCurrencyCodes() : AllCurrencyResponse

RPC API + DTO + Request/Response

Name Operation	Input	Output	Description
getExchangeRate	**ExchangeRateRequest** = **<from: CurrencyCode, to: CurrencyCode>**	**ExchangeRateResponse** = **<from: CurrencyCode, to: CurrencyCode, rate: Float, date: DateTime>**	Returns the exchange rate between two given currencies.
getAllExchangeRates		**AllExchangeRatesResponse** = Set(**ExchangeRateResponse**)	Returns the exchange rates between every two currencies.
getAllCurrencyCodes		**AllCurrencyResponse** = Set (currencyCode:CurrencyCode)	Returns all the currency codes

DTOs

RPC API + DTO + Request/Acknowlege::Poll

Let us suppose that the operation **getAllExchangeRates** takes some time to be processed.

Name Operation	Input	Output	Description
getAllExchangeRates		**getAllExchangeRatesACK = <id: String>**	Returns the acknowledgment that the request to obtain the exchange rates between every two currencies will be processed.
getResult	**getAllExchangeRatesACK = <id: String>**	AllExchangeRatesResponse = Set(**ExchangeRateResponse**)	Returns the result of the previous request

DTO

References

- Robert Daigneau. **Service Design Patterns. Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services.** Addison Wesley, 2012
- https://www.w3schools.com/xml/xml_services.asp