

arnaud.nauwynck@gmail.com

Part 3 : Api, Protocol Marshalling, http Rest Api

Generalizing http Rest Json, principles for better code architecture

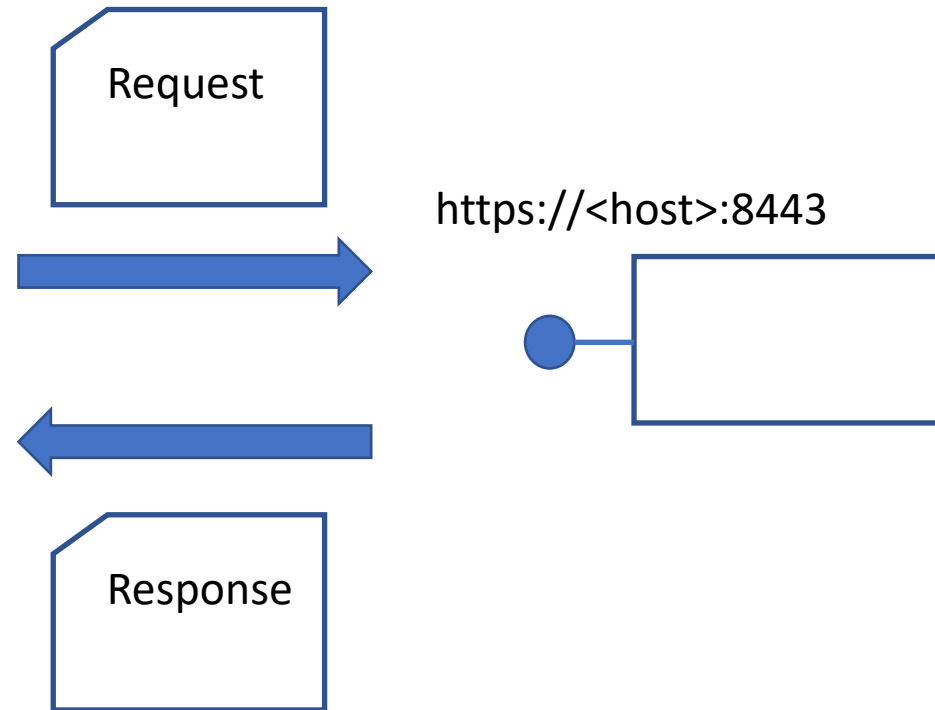
This document:

[http://github.com/arnaud-nauwynck/Presentations/java/
ApiProtocol.pdf](http://github.com/arnaud-nauwynck/Presentations/java/ApiProtocol.pdf)

Api http Server ... for http Clients



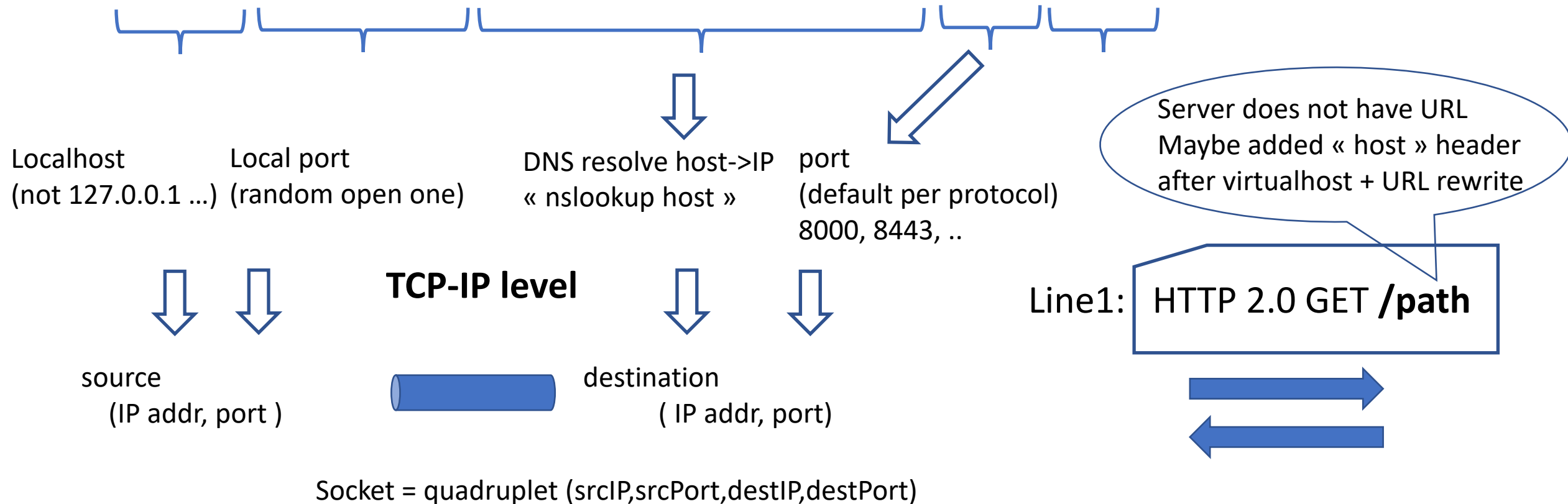
curl://



http Call: VERB + path + headers + body

« URL » = host + port + Path

https://user:pass@somehost.some.domain:8443/path



Remote « Call »

asymmetric socket read-write

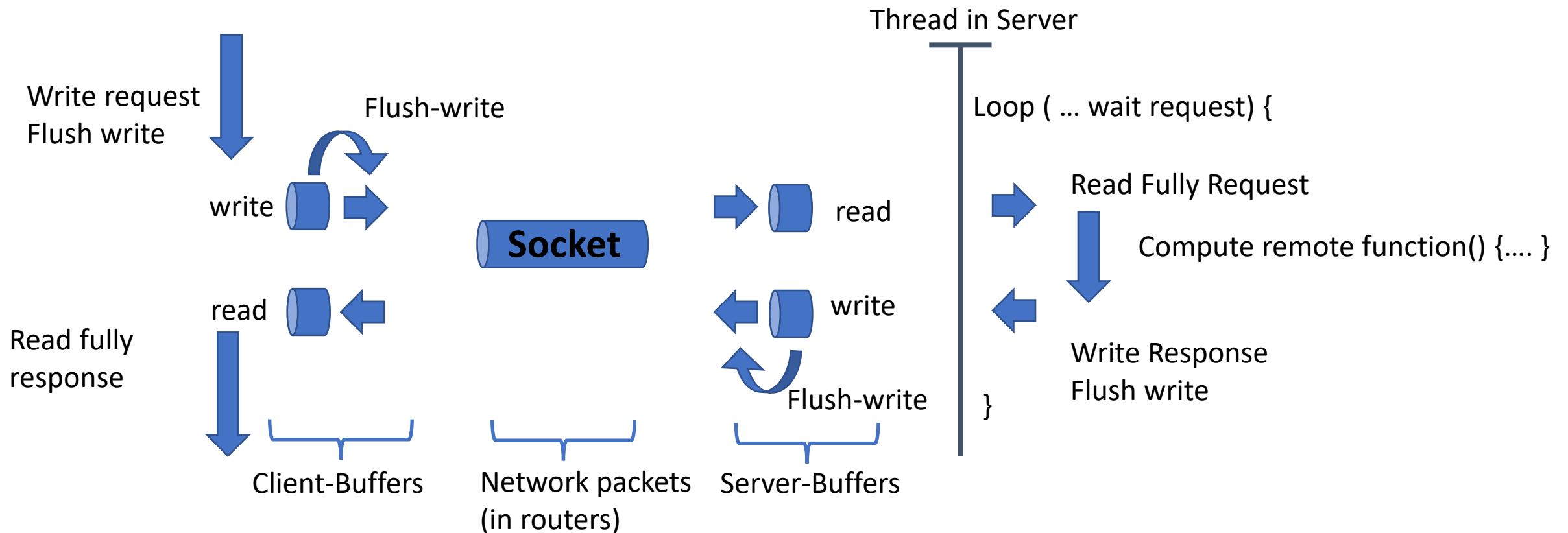
active Client – waiting Server, Processing

On Client-side:

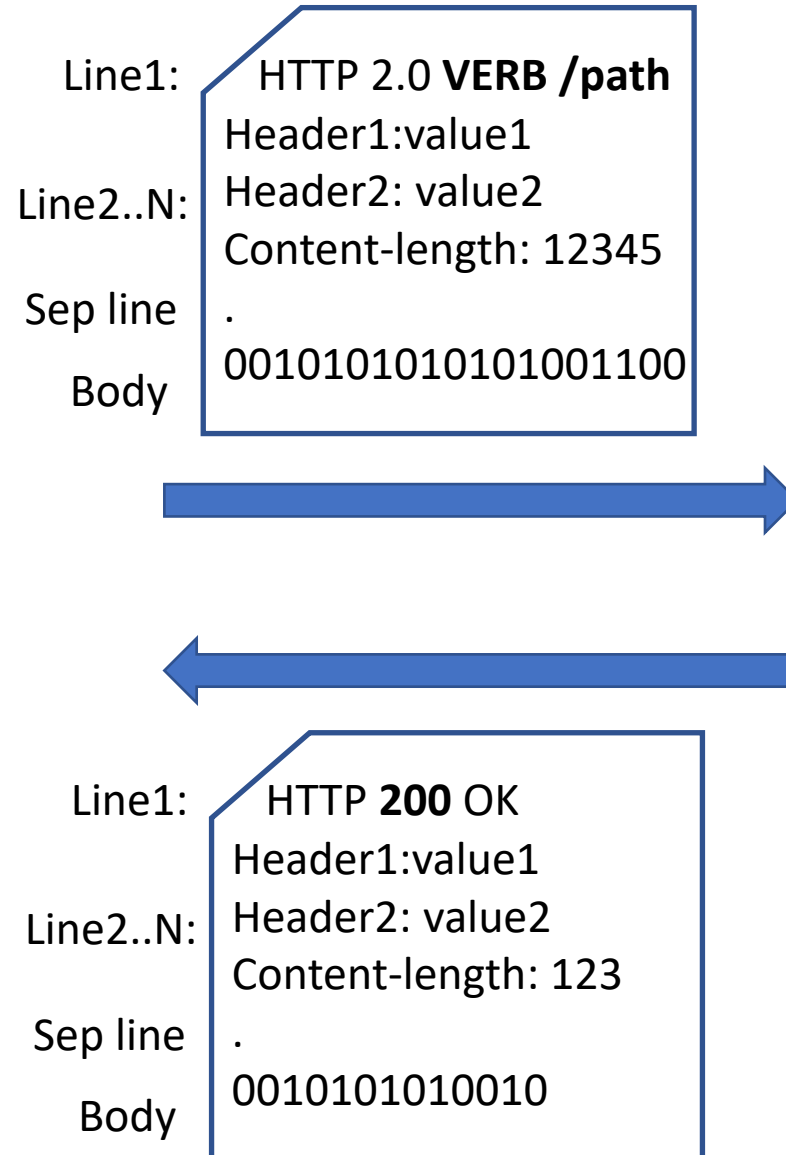
Write,flush-write ... then (blocking-wait) read

On Server:

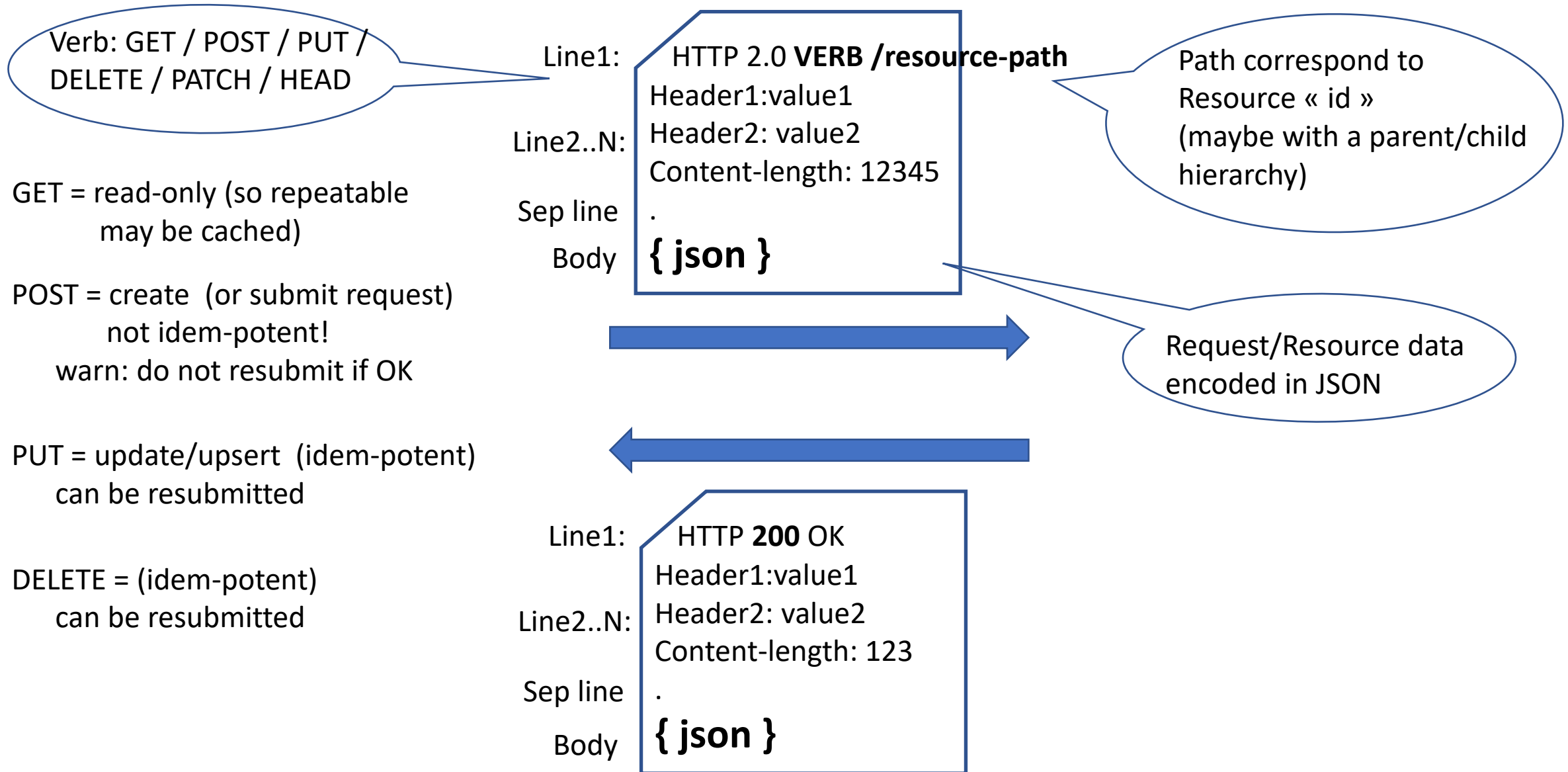
(blocking-wait) read ... then write,flush-write



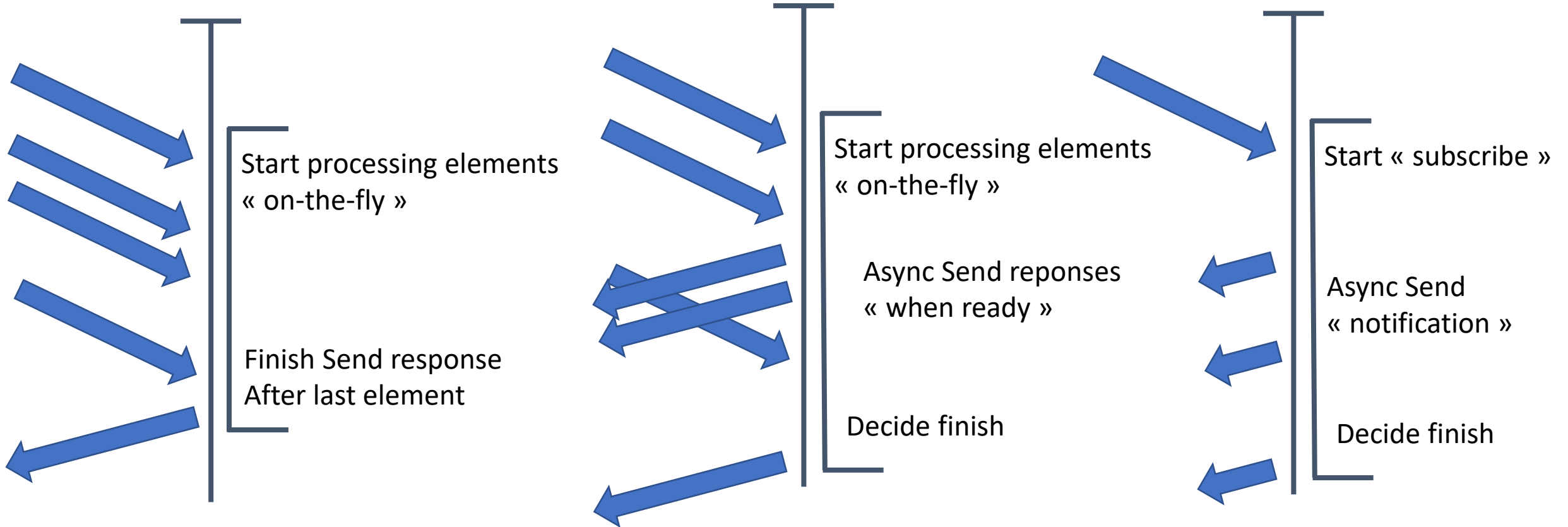
Simple Http Request Model



Rest (http + Json) Requests



Other Asynchronous Call Models



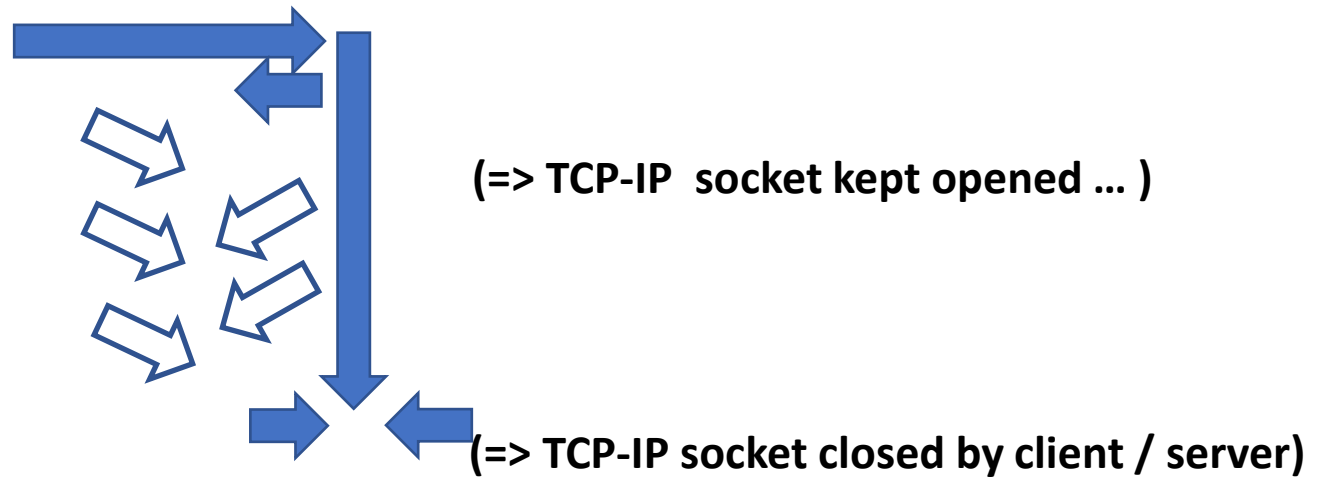
Example gRPC (over Https..)

```
service Greeter {  
  
    // send 1 request => receive 1 response  
    rpc SayHello (HelloRequest) returns (HelloReply);  
  
    // send 1 request => receive * responses stream  
    rpc SayHello_OutputStream (HelloRequest) returns (stream HelloReply);  
  
    // send * request stream => 1 response (wait X request, then respond)  
    rpc SayHello_InStream (stream HelloRequest) returns (HelloReply);  
  
    // send * request stream => receive * responses stream  
    rpc SayHello_InStream_OutputStream (stream HelloRequest) returns (stream HelloReply);  
  
}  
  
message HelloRequest {  
    string name = 1;  
}  
  
message HelloReply {  
    string message = 1;  
}
```


WebSocket

(ws:// instead of https://)

ws://somehost.somedomain

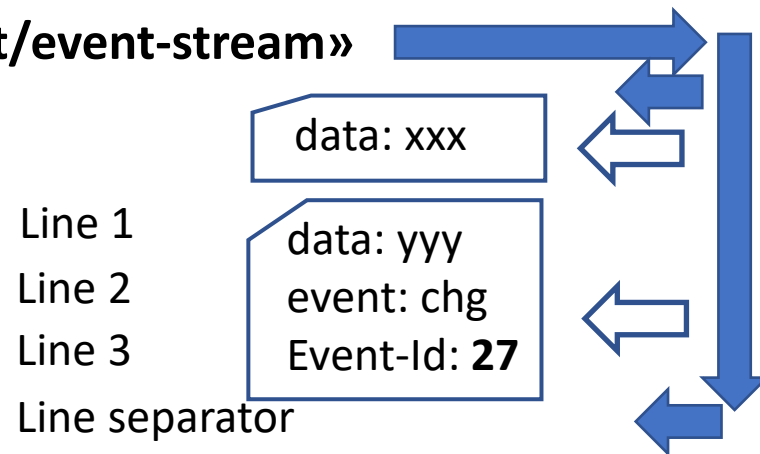


SSE : Server – Sent – Event

+ still https:// with socket negotiation
« medium wait » Polling by client

http GET /sse

Accept: « text/event-stream »



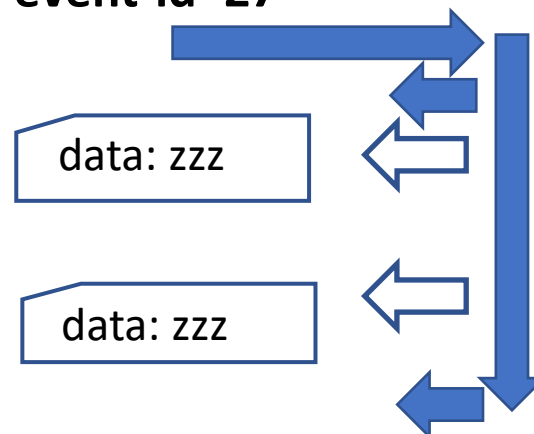
HTTP 200 OK

Content-type: **text/event-stream**

(=> **socket kept opened ...**)

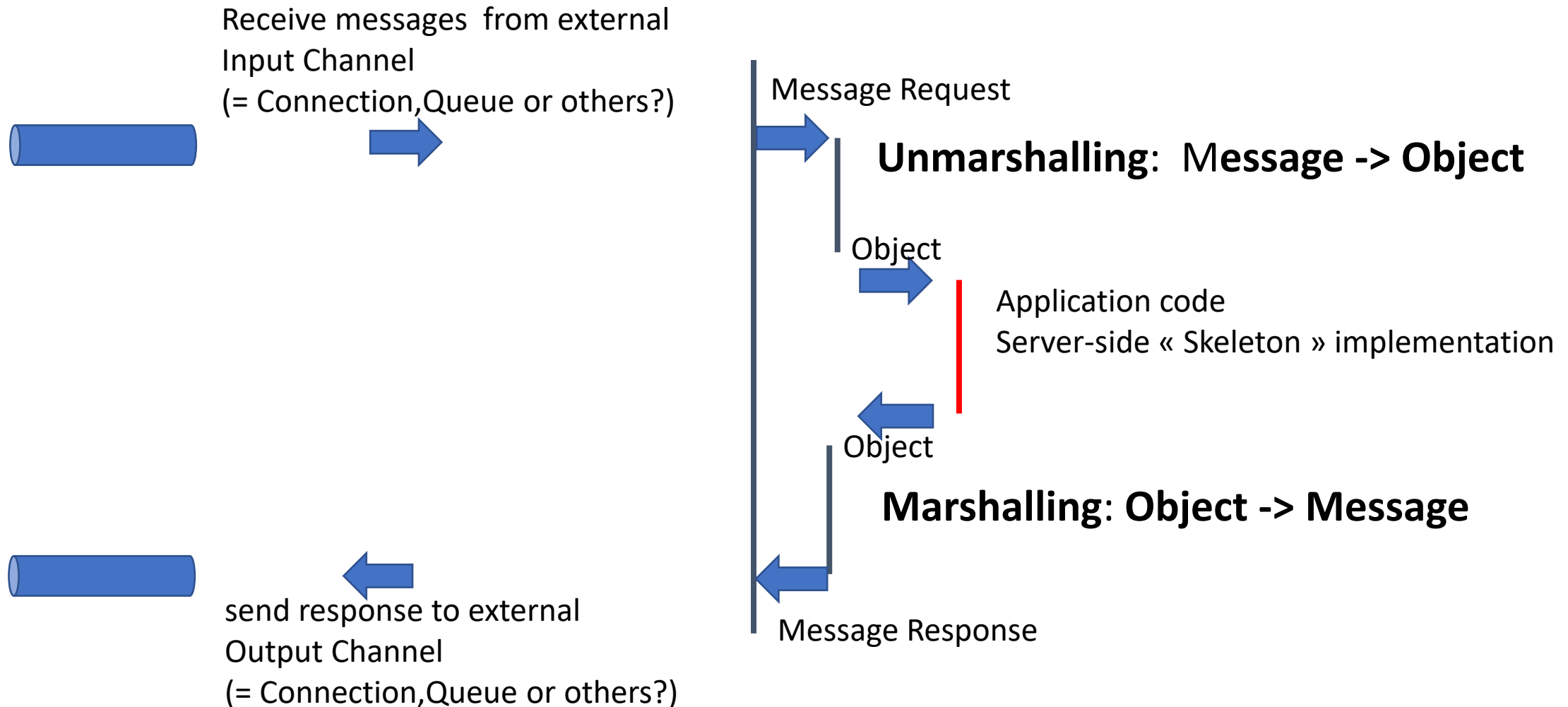
Timeout ... response / close connection

http GET /sse?**last-event-id=27**

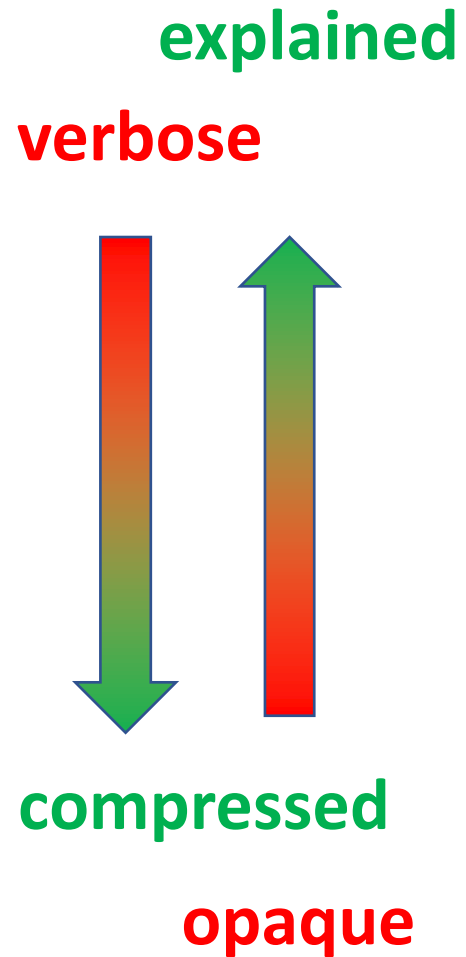


External Protocol

... Marshalling/Unmarshalling to Internal Language



Message Encoding : Text / Binary



<xml? schema:=« ..xsd » namespace=« ns1:..... » >
 <a> <ns1:b> some value</ns1:b>

+++ most powerfull
+++ Self-explained / extensible
--- most verbose

<xml?> <a> some value

JSON: { « a »: { « b »: « some value » } }

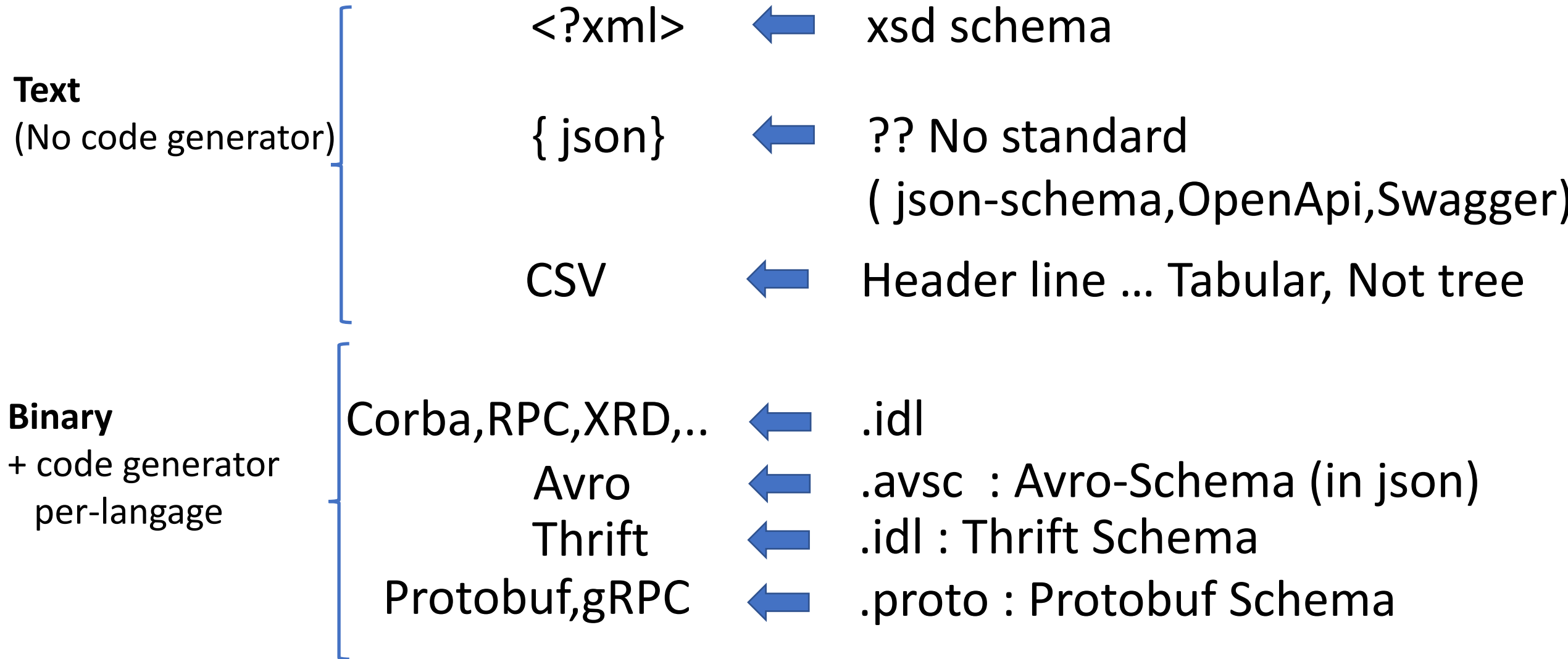
Simplest
Compromise for Web

CSV: a.b; \n
 some value

Binary: 010101010110101010101010

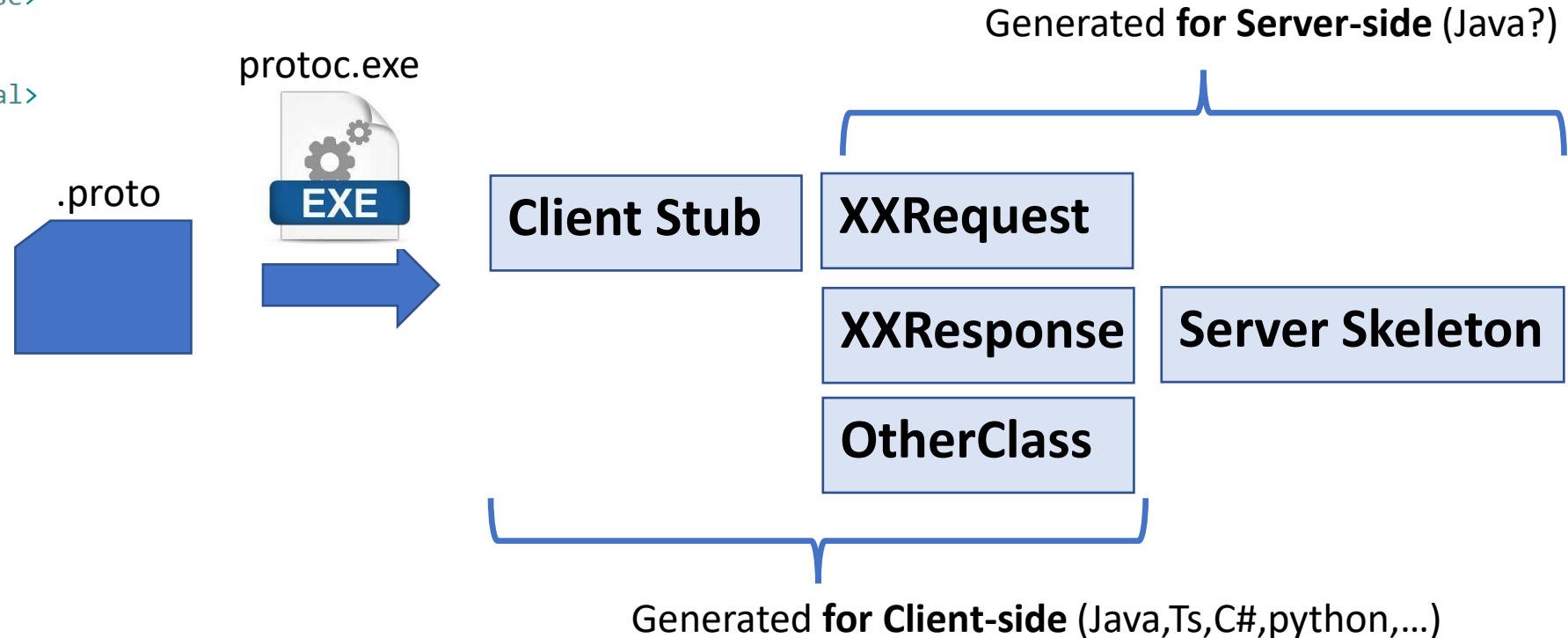
Obscure
+++ compressed encoding
+++ efficient int32, long64, float32, ..
CPU representation

Schema ... Code-Generator



Example Code-Generator: gPRC (protobuf-maven-plugin)

```
<plugin>
  <groupId>org.xolstice.maven.plugins</groupId>
  <artifactId>protobuf-maven-plugin</artifactId>
  <version>0.6.1</version>
  <configuration>
    <protocArtifact>com.google.protobuf:protoc:${protoc.version}:exe:${os.detected.classifier}</protocArtifact>
    <pluginId>grpc-java</pluginId>
    <pluginArtifact>io.grpc:protoc-gen-grpc-java:${grpc.version}:exe:${os.detected.classifier}</pluginArtifact>
  </configuration>
  <executions>
    <execution>
      <phase>generate-sources</phase>
      <goals>
        <goal>compile</goal>
        <goal>compile-custom</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```



Generated Code: POJO + Builder + writeTo(Out..) / parseFrom(In..)

```
message HelloMessage {  
  
    string  fieldString1  = 1;  
  
    int32   fieldInt2     = 2;  
    int64   fieldLong3    = 3;  
  
    float   fieldFloat4   = 4;  
    double  fieldDouble5  = 5;  
  
    bool    fieldBool6    = 6;  
  
}
```

protoc.exe



```
    SF FIELDSTRING1_FIELD_NUMBER : int  
    V fieldString1_ : Object  
    A getFieldString1() : String  
    A getFieldString1Bytes() : ByteString  
    SF FIELDINT2_FIELD_NUMBER : int  
    I fieldInt2_ : int  
    A getFieldInt2() : int  
    SF FIELDLONG3_FIELD_NUMBER : int  
    L fieldLong3_ : long  
    A getFieldLong3() : long  
    SF FIELDFLOAT4_FIELD_NUMBER : int  
    F fieldFloat4_ : float  
    A getFieldFloat4() : float  
    SF FIELDDOUBLE5_FIELD_NUMBER : int  
    D fieldDouble5_ : double  
    A getFieldDouble5() : double  
    SF FIELDBOOL6_FIELD_NUMBER : int  
    B fieldBool6_ : boolean  
    A getFieldBool6() : boolean  
  
    A newBuilderForType() : Builder  
    S newBuilder() : Builder  
    S newBuilder(HelloMessage) : Builder  
    A toBuilder() : Builder  
    GF Builder  
        SF getDescriptor() : Descriptor  
        A clear() : Builder  
        A getDescriptorForType() : Descriptor  
        A getDefaultInstanceForType() : HelloMessage  
        A build() : HelloMessage  
        A buildPartial() : HelloMessage  
        A clone() : Builder  
        A setField(FieldDescriptor, Object) : Builder
```

```
    A writeTo(CodedOutputStream) : void  
    A getSerializedSize() : int  
  
    S parseFrom(ByteBuffer) : HelloMessage  
    S parseFrom(ByteBuffer, ExtensionRegistryLite) : HelloMessage  
    S parseFrom(ByteString) : HelloMessage  
    S parseFrom(ByteString, ExtensionRegistryLite) : HelloMessage  
    S parseFrom(byte[]) : HelloMessage  
    S parseFrom(byte[], ExtensionRegistryLite) : HelloMessage  
    S parseFrom(InputStream) : HelloMessage  
    S parseFrom(InputStream, ExtensionRegistryLite) : HelloMessage  
    S parseDelimitedFrom(InputStream) : HelloMessage  
    S parseDelimitedFrom(InputStream, ExtensionRegistryLite) : HelloMessage  
    S parseFrom(CodedInputStream) : HelloMessage  
    S parseFrom(CodedInputStream, ExtensionRegistryLite) : HelloMessage
```

writeTo() = efficient binary Marshalling
parseFrom() = .. Un-Marshalling

```
@java.lang.Override
public void writeTo(com.google.protobuf.CodedOutputStream output)
    throws java.io.IOException {
    if (!getFieldString1Bytes().isEmpty()) {
        com.google.protobuf.GeneratedMessageV3.writeString(output, 1, fieldString1_);
    }
    if (fieldInt2_ != 0) {
        output.writeInt32(2, fieldInt2_);
    }
    if (fieldLong3_ != 0L) {
        output.writeInt64(3, fieldLong3_);
    }
    if (fieldFloat4_ != 0F) {
        output.writeFloat(4, fieldFloat4_);
    }
    if (fieldDouble5_ != 0D) {
        output.writeDouble(5, fieldDouble5_);
    }
    if (fieldBool6_ != false) {
        output.writeBool(6, fieldBool6_);
    }
    unknownFields.writeTo(output);
}
```


Sample Server Implementation

```
import fr.an.tests.testpgRPC.GreeterGrpc;
import fr.an.tests.testpgRPC.HelloReply;
import fr.an.tests.testpgRPC.HelloRequest;

import io.grpc.stub.StreamObserver;

public final class GrpcImplService extends GreeterGrpc.GreeterImplBase {

    @Override
    public void sayHello(HelloRequest req, StreamObserver<HelloReply> responseObserver) {
        String reqName = req.getName();
        Log.info("sayHello req.name:" + reqName);
        HelloReply reply = HelloReply.newBuilder().setMessage("Hello " + reqName + ", from " + serverName).build();

        Log.info(".. response (onNext+onComplete) sayHello req.name:" + reqName);
        responseObserver.onNext(reply);
        responseObserver.onCompleted();
    }

    @Override
    public void sayHelloOutputStream(HelloRequest req, StreamObserver<HelloReply> responseObserver) {
        String reqName = req.getName();
        Log.info("sayHello req.name:" + reqName);
        HelloReply reply = HelloReply.newBuilder().setMessage("Hello " + reqName + ", from " + serverName).build();

        Log.info(".. response 5x onNext + 5x (sleep + onNext) + onComplete) sayHello req.name:" + reqName);
        for(int i = 0; i < 5; i++) {
            responseObserver.onNext(reply);
        }
        for(int i = 0; i < 5; i++) {
            sleep(1000);
            responseObserver.onNext(reply);
        }

        responseObserver.onCompleted();
    }
}
```

Sample Client Calls

```
Channel channel = ManagedChannelBuilder.forAddress(host, port).usePlaintext().build();
this.blockingStub = GreeterGrpc.newBlockingStub(channel);
this.asyncStub = GreeterGrpc.newFutureStub(channel);
this.stub = GreeterGrpc.newStub(channel);
Log.info("call to channel.. " + channel);
}

public void sayHello_blockingStub() throws Exception {
    HelloRequest helloReq = HelloRequest.newBuilder().setName("you").build();

    HelloReply reply = blockingStub.sayHello(helloReq);

    Log.info("sayHello => " + reply.getMessage());
}
```

gRPC ... Async for performance

(see also java concurrent.Future / ReactiveJava / ProjectReactor ...)

Details

```
package io.grpc.stub;
```

```
    * Receives notifications from an observable stream of messages.[]
```

```
public interface StreamObserver<V> {
```

```
    * Receives a value from the stream.[]
```

```
    void onNext(V value);
```

```
    * Receives a terminating error from the stream.[]
```

```
    void onError(Throwable t);
```

```
    * Receives a notification of successful stream completion.[]
```

```
    void onCompleted();
```

```
}
```

Sample Client Calls... Async

```
public void sayHello_asyncStub_get() throws Exception {
    HelloRequest helloReq = HelloRequest.newBuilder().setName("you").build();
    ListenableFuture<HelloReply> resListenableFuture = asyncStub.sayHello(helloReq);

    // wrap in java.util.CompletableFuture, use .thenApply(), thenCombine(), ...
    CompletableFuture<HelloReply> completableFuture = MoreFutures.toCompletableFuture(resListenableFuture);
    CompletableFuture<HelloReply> resFuture = completableFuture// .thenCombine(..) // combine with other futures
        .thenApply(x -> x); // transform when ready

    HelloReply reply = resFuture.get(); // (generally don't do that) async to blocking!
    Log.info("sayHello (async block) => " + reply.getMessage());
}

public void sayHello_stub() throws Exception {
    HelloRequest helloReq = HelloRequest.newBuilder().setName("you").build();
    CountdownLatch latch = new CountdownLatch(1);
    stub.sayHello(helloReq, new StreamObserver<HelloReply>() {
        @Override
        public void onNext(HelloReply reply) {
            Log.info("sayHello .. response => " + reply.getMessage());
        }
        @Override
        public void onCompleted() {
            Log.info("sayHello .. onComplete");
            latch.countDown();
        }
        @Override
        public void onError(Throwable t) {
            Log.info("sayHello .. onError", t);
            latch.countDown();
        }
    });
    latch.await();
    Log.info(".. done wait sayHello");
}
```

Details

Generalizing gRPC « pattern skeleton code » (call Service ... Entity to DTO ... to Message)

```
@Autowired
FooService delegateXAService;

@Override
public void sayHello(HelloRequest req, StreamObserver<HelloReply> responseObserver) {
    // Step 1/3:
    // validity check, extract/convert gRPC Request to internal classes (internal DTO)
    long startTime = System.currentTimeMillis();
    String reqName = req.getName();
    Log.info("sayHello name:" + reqName);
    FooDTO inDto = new FooDTO(reqName);

    // Step 2/3:
    // delegate to transactional service (SOLID principle)
    FooDTO tmpres = delegateXAService.saveHello(inDto);

    // Step 3/3:
    // convert internal DTO to gRPC response, marshall response
    long millis = System.currentTimeMillis() - startTime;
    Log.info(".. done sayHello, took " + millis);
    HelloReply reply = HelloReply.newBuilder().setMessage("Hello " + tmpres.message).build();
    responseObserver.onNext(reply);
    responseObserver.onCompleted();
}
```

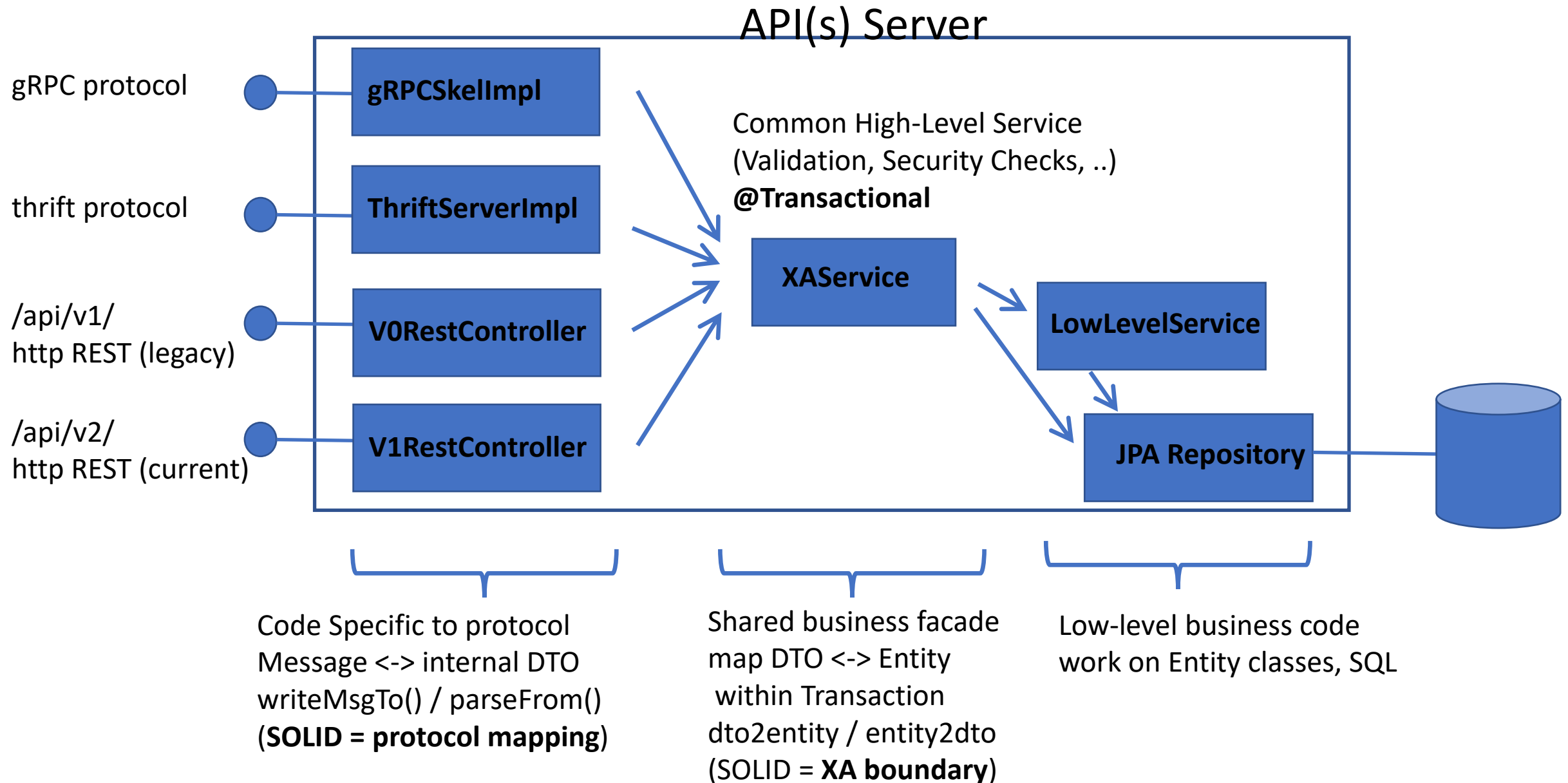
Generalizing Server-side « pattern skeleton code »

```
@XXXProtocolMapping(path="/sayHello")
public void sayHello(
    @XXXRequestBody HelloRequest req,
    @XXXResponse XXXOutputResponse out) {
    // Step 1/3:
    // validity check, extract/convert XXX Request to internal classes (internal DTO)
    long startTime = System.currentTimeMillis();
    String reqName = req.getName();
    Log.info("sayHello name:" + reqName);
    FooDTO inDto = new FooDTO(reqName);

    // Step 2/3:
    // delegate to transactional service (SOLID principle)
    FooDTO tmpres = delegateXAService.saveHello(inDto);

    // Step 3/3:
    // convert internal DTO to XXX response, marshall response
    long millis = System.currentTimeMillis() - startTime;
    Log.info(".. done sayHello, took " + millis);
    HelloReply reply = HelloReply.newBuilder().setMessage("Hello " + tmpres.message).build();
    out.status(200).addHeader("header1", "value1").body(reply);
}
```

Design code for several protocols / versionned APIs



gRPC / protobuf
... designed for backward compatibility

Zooming on backward compatibility
... ideas for Rest / Json backward compatibility ?

Schema: Fixed / support version upgrade

Backward-compatibility is mandatory

Still evolutivity possible ? Better if true



Dynamic Schema: Protobuf, gRPC compiled Versioned Fields

In schema, all fields are numbered (unique ID)

```
message HelloRequest {  
  string name = 1;  
}
```

```
message HelloRequest {  
  string name = 1;  
  string color = 2; // added in version 2  
}
```

```
message HelloRequest {  
  string name = 1;  
  string color = 2; // deprecated, unused in version >=3 .. replaced by cssStyle  
  string cssStyle = 3;  
}
```

Dynamic Messages, Schema-compatible

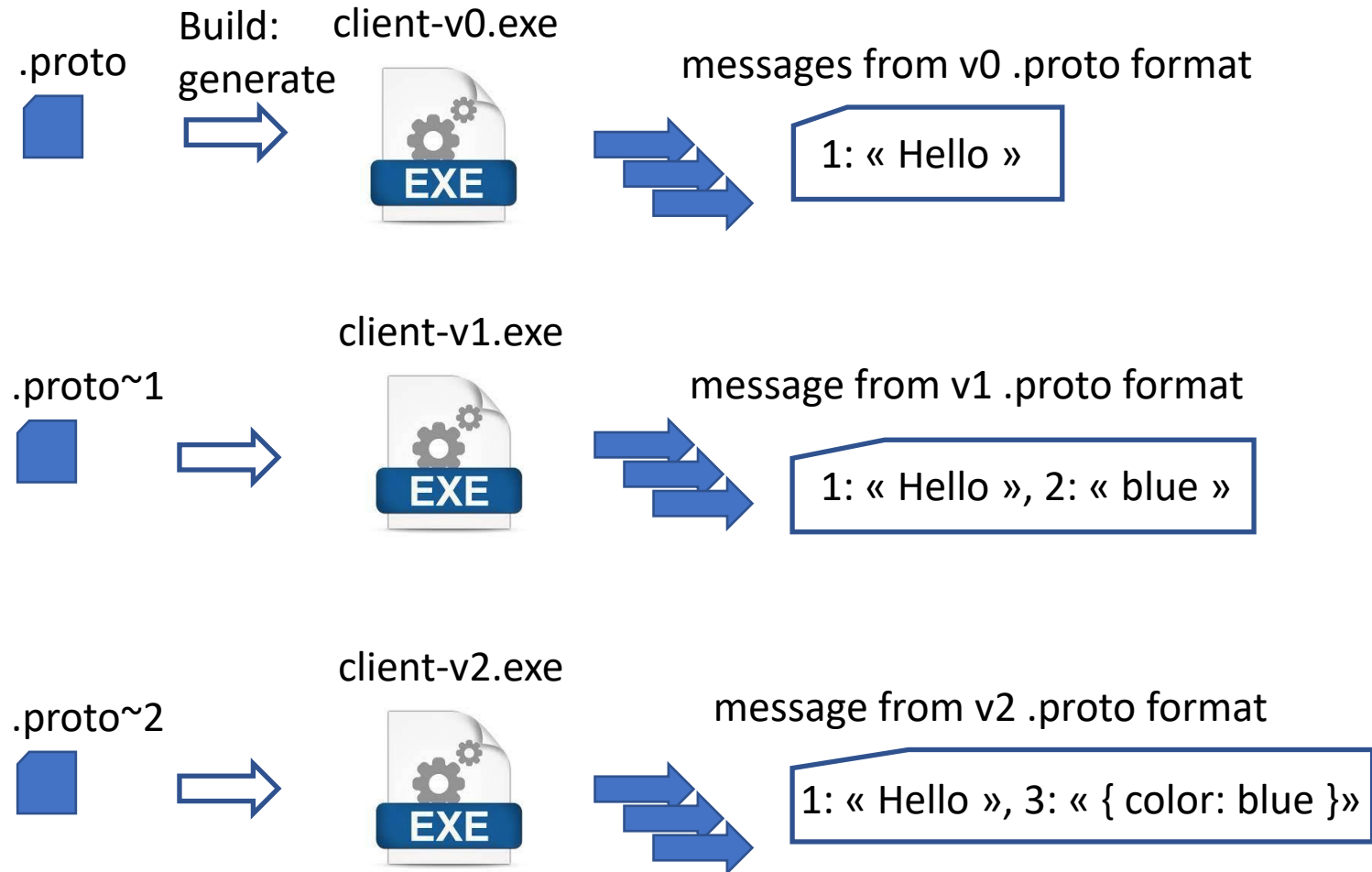
In Messages

« Map<FieldId, Value> »

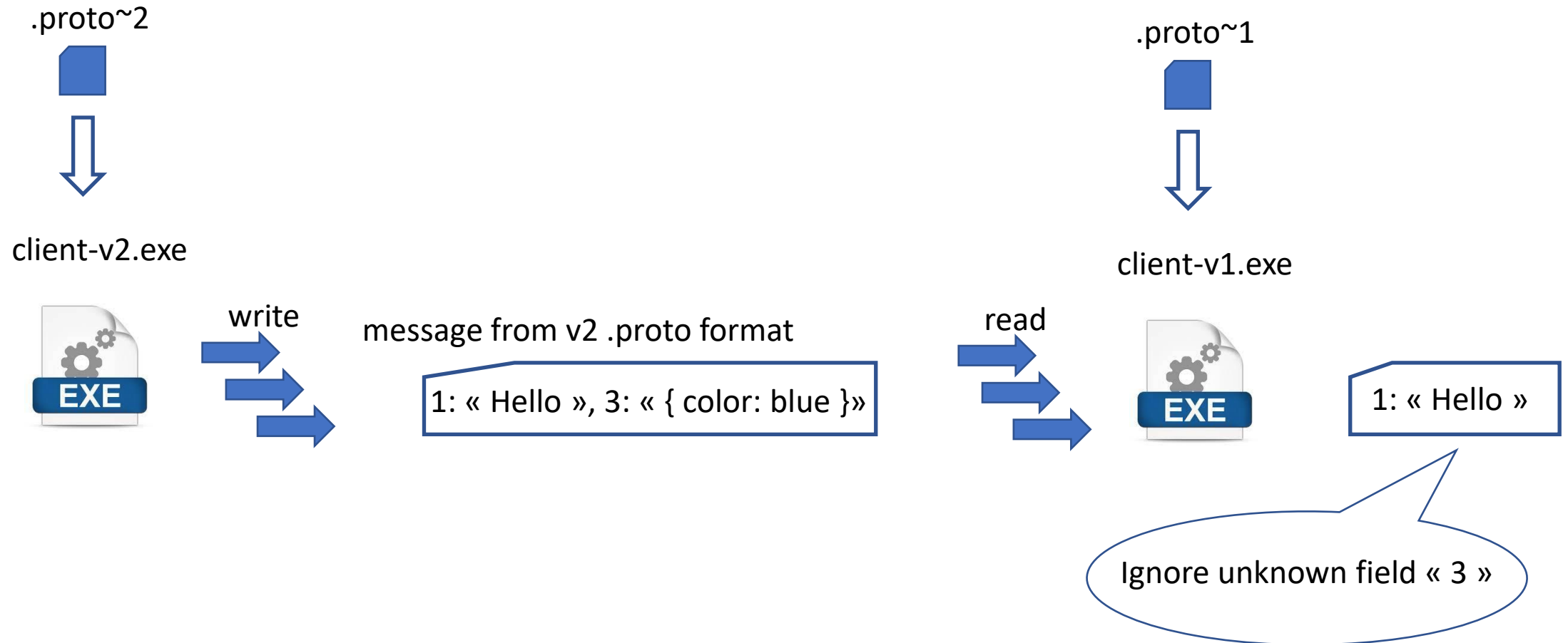
fieldId looks « unnecessary »

Small extra cost

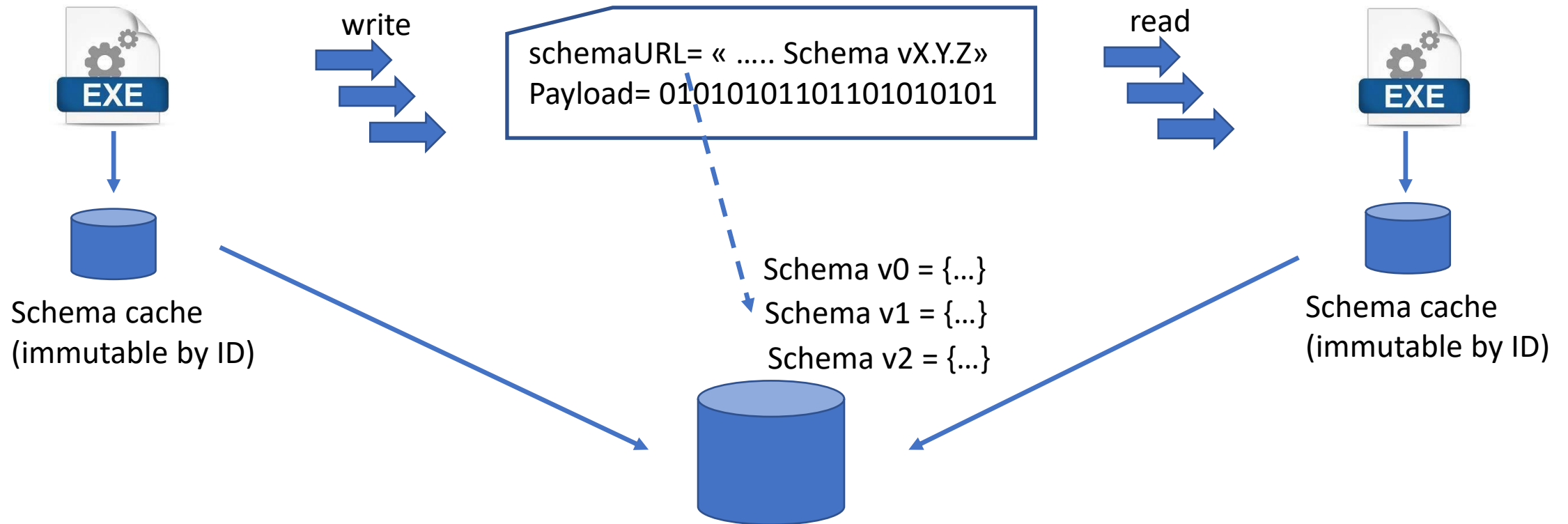
... Great compatibility



Reading « Future » Message / Backward Client ... Ignore Unknown Fields



Schema Registry



Schema Contained in Messages

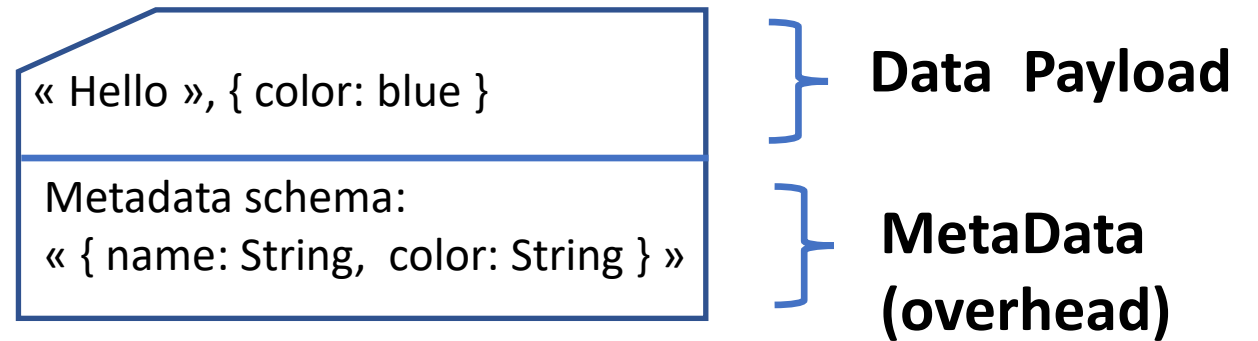
Examples :

Kafka Messages, Pulsar Messages

Avro Message, Avro Data-File

PARQUET File for ULTRA compression of millions of rows (dictionary, incremental, filter..)

java.io.Serializable Contains serialVersionUID + fully-qualified Names + field name / types
Use « Kryo » instead of « java.io. » for typed / schema-less messages !
(better performances)



Documenting Rest APIs

Exposing Rest Api « Metadata »

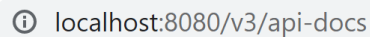


Springdoc-openapi-ui


```
<dependency>  
  <groupId>org.springdoc</groupId>  
  <artifactId>springdoc-openapi-ui</artifactId>  
  <version>${springdoc-openapi-ui.version}</version>  
</dependency>
```


JSON api-docs

<http://localhost:8080/v3/api-docs>

[illegible]

http://localhost:8080/swagger-ui.html

 **Swagger**
Supported by SMARTBEAR

/v3/api-docs

Explore

OpenAPI definition

v0 OAS3

/v3/api-docs

Servers

http://localhost:8080 - Generated server url

v-4-todo-rest-controller

GET

/api/v4/todo

⌵

PUT

/api/v4/todo

⌵

POST

/api/v4/todo

⌵

GET

/api/v4/todo/{id}

⌵

DELETE

/api/v4/todo/{id}

⌵

Detailed Rest operation

PUT

/api/v4/todo

Parameters

Try it out

No parameters

Request body required

application/json

Example Value | Schema

```
{  "id": 0,  "label": "string",  "priority": 0}
```

Responses

Code	Description	Links
200	<div>OK</div> <div>Media type<div>*/*</div></div> <div>Controls Accept header.</div> <div>Example Value Schema<div><pre>{ "id": 0, "label": "string", "priority": 0}</pre></div></div>	No links

Execute Query

GET

/api/v4/todo

^

Parameters

Cancel

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/api/v4/todo' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/api/v4/todo
```

Server response

Code

Details

200

Response body

```
[
  {
    "id": 1,
    "label": "Apprendre Maven",
    "priority": 0
  },
  {
    "id": 2,
    "label": "Apprendre Spring-boot",
    "priority": 0
  }
]
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Sun,30 Jan 2022 21:10:59 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code

Description

Links

200

OK

No links

Media type

/

Controls Accept header.

Example PUT with Json RequestBody

PUT

/api/v4/todo

Parameters

No parameters

Request body required

application/json

```
{  "label": "learn Swagger - OpenApi",  "priority": 3}
```

Execute

Responses

Code	Description	Links
200	OK	No links

Media type

/

Controls Accept header.

Example Value | Schema

```
{  "id": 0,  "label": "string",  "priority": 0}
```

<https://github.com/swagger-api/swagger-codegen>

This is the Swagger Codegen project, which allows generation of API client libraries (SDK generation), server stubs and documentation automatically given an [OpenAPI Spec](#). Currently, the following languages/frameworks are supported:

- **API clients:** **ActionScript**, **Ada**, **Apex**, **Bash**, **C#** (.net 2.0, 3.5 or later), **C++** (cpprest, Qt5, Tizen), **Clojure**, **Dart**, **Elixir**, **Elm**, **Eiffel**, **Erlang**, **Go**, **Groovy**, **Haskell** (http-client, Servant), **Java** (Jersey1.x, Jersey2.x, OkHttp, Retrofit1.x, Retrofit2.x, Feign, RestTemplate, RESTEasy, Vertx, Google API Client Library for Java, Rest-assured), **Kotlin**, **Lua**, **Node.js** (ES5, ES6, AngularJS with Google Closure Compiler annotations) **Objective-C**, **Perl**, **PHP**, **PowerShell**, **Python**, **R**, **Ruby**, **Rust** (rust, rust-server), **Scala** (akka, http4s, swagger-async-httpclient), **Swift** (2.x, 3.x, 4.x, 5.x), **Typescript** (Angular1.x, Angular2.x, Fetch, jQuery, Node)
- **Server stubs:** **Ada**, **C#** (ASP.NET Core, NancyFx), **C++** (Pistache, Restbed), **Erlang**, **Go**, **Haskell** (Servant), **Java** (MSF4J, Spring, Undertow, JAX-RS: CDI, CXF, Inflector, RestEasy, Play Framework, [PKMST](#)), **Kotlin**, **PHP** (Lumen, Slim, Silex, [Symfony](#), [Zend Expressive](#)), **Python** (Flask), **NodeJS**, **Ruby** (Sinatra, Rails5), **Rust** (rust-server), **Scala** ([Finch](#), [Lagom](#), Scalatra)
- **API documentation generators:** **HTML**, **Confluence Wiki**
- **Configuration files:** [Apache2](#)
- **Others:** **JMeter**

Swagger CodeGen maven-plugin

```
<plugin>
  <groupId>io.swagger.codegen.v3</groupId>
  <artifactId>swagger-codegen-maven-plugin</artifactId>
  <version>${swagger-codegen.version}</version>

  <configuration>
    <inputSpec>http://localhost:8080/v3/api-docs</inputSpec>
  </configuration>
  <executions>
    <execution>
      <id>generate-swagger-typescript-angular-7</id>
      <phase>generate-sources</phase>
      <goals>
        <goal>generate</goal>
      </goals>
      <configuration>
        <language>typescript-angular</language>
        <output>${basedir}/target/generated-typescript-angular7</output>
        <configOptions>
          <ngVersion>7.2.12</ngVersion>
        </configOptions>
      </configuration>
    </execution>
  </executions>
</plugin>
```



Example: Generate
typescript Angular

mvn generate-sources

```
$ mvn -Pswagger-gen generate-sources
[INFO] Scanning for projects...
[INFO]
[INFO] -----< fr.an.tests:test-springboot-cruds >-----
[INFO] Building demo 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- swagger-codegen-maven-plugin:3.0.32:generate (generate-swagger-typescript-angular-7) @ test-springboot-cruds ---
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\model\todoDTO.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\api\todoRestController.service.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\model\models.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\api\api.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\index.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\api.module.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\configuration.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\variables.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\encoder.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\.gitignore
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\.npmignore
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\git_push.sh
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\ng-package.json
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\.swagger-codegen\VERSION
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.609 s
[INFO] Finished at: 2022-01-30T22:21:21+01:00
[INFO] -----
```


Generated Code (example: Typescript -Angular)

```
/**
 * OpenAPI definition
 * No description provided (generated by Swagger Codegen https://github.com/swagger-api/swagger-codegen)
 *
 * OpenAPI spec version: v0
 *
 * NOTE: This class is auto generated by the swagger code generator program.
 * https://github.com/swagger-api/swagger-codegen.git
 * Do not edit the class manually.
 */
import { Inject, Injectable, Optional } from '@angular/core';
import { HttpClient, HttpHeaders, HttpParams,
       HttpResponse, HttpEvent } from '@angular/common/http';
import { CustomHttpUrlEncodingCodec } from '../encoder';

import { Observable } from 'rxjs';

import { ResponseDTO } from '../model/responseDTO';
import { TodoDTO } from '../model/todoDTO';

import { BASE_PATH, COLLECTION_FORMATS } from '../variables';
import { Configuration } from '../configuration';

@Injectable()
export class TodoRestControllerService {

    protected basePath = 'http://localhost:8080';
    public defaultHeaders = new HttpHeaders();
    public configuration = new Configuration();

    constructor(protected httpClient: HttpClient, @Optional()@Inject(BASE_PATH) basePath: string, @Optional() confi
        if (basePath) {
            this.basePath = basePath;
        }
        if (configuration) {
            this.configuration = configuration;
            this.basePath = basePath || configuration.basePath || this.basePath;
        }
    }

    public putTodo4(body: TodoDTO, observe?: 'body', reportProgress?: boolean): Observable<TodoDTO>;
    public putTodo4(body: TodoDTO, observe?: 'response', reportProgress?: boolean): Observable<HttpResponse<TodoDTO>>;
    public putTodo4(body: TodoDTO, observe?: 'events', reportProgress?: boolean): Observable<HttpEvent<TodoDTO>>;
    public putTodo4(body: TodoDTO, observe: any = 'body', reportProgress: boolean = false ): Observable<any> {

        if (body === null || body === undefined) {
            throw new Error('Required parameter body was null or undefined when calling putTodo4.');
```

Another Example : generate Java

```
<plugin>
  <groupId>io.swagger.codegen.v3</groupId>
  <artifactId>swagger-codegen-maven-plugin</artifactId>
  <version>${swagger-codegen.version}</version>
  <configuration>
    <inputSpec>http://localhost:8080/v3/api-docs</inputSpec>
  </configuration>
  <executions>
    <execution>
      <id>generate-java</id>
      <phase>generate-sources</phase>
      <goals>
        <goal>generate</goal>
      </goals>
      <configuration>
        <language>java</language>
        <output>${basedir}/target/generated-java</output>
      </configuration>
    </execution>
    <execution>
      <id>generate-java-retrofit2</id>
      <phase>generate-sources</phase>
      <goals>
        <goal>generate</goal>
      </goals>
      <configuration>
        <language>java</language>
        <library>retrofit2</library>
        <output>${basedir}/target/generated-java-retrofit2</output>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Java
Default library=OkHttp

Java
+ library=Retrofit2

Generated Code (Java, default using OkHttp)

```
package io.swagger.client.api;
```

```
import io.swagger.client.ApiCallback;
```

```
public class TodoRestControllerApi {  
    private ApiClient apiClient;  
  
    public TodoRestControllerApi() {  
        this(Configuration.getDefaultApiClient());  
    }
```

```
    public TodoRestControllerApi(ApiClient apiClient) {  
        this.apiClient = apiClient;  
    }
```

```
    * Build call for putTodo4  
    public com.squareup.okhttp.Call putTodo4Call(TodoDTO body, final ProgressResponseBody.ProgressListener progress  
        Object localVarPostBody = body;  
  
        / create path and map variables  
        String localVarPath = "/api/todo";  
  
        List        List  
        Map<String, String> localVarHeaderParams = new HashMap<String, String>();  
  
        Map<String, Object> localVarFormParams = new HashMap<String, Object>();  
  
        List<String> localVarAccepts = {  
            "*"/*  
        };  
        List<String> localVarAccept = apiClient.selectHeaderAccept(localVarAccepts);  
        if (localVarAccept != null) localVarHeaderParams.put("Accept", localVarAccept);  
  
        List<String> localVarContentTypes = {  
            "application/json"  
        };  
        List<String> localVarContentType = apiClient.selectHeaderContentType(localVarContentTypes);  
        localVarHeaderParams.put("Content-Type", localVarContentType);  
  
        if (progressListener != null) {  
            apiClient.getHttpClient().networkInterceptors().add(new com.squareup.okhttp.Interceptor() {  
                @Override  
                public com.squareup.okhttp.Response intercept(com.squareup.okhttp.Interceptor.Chain chain) throws I  
                    com.squareup.okhttp.Response originalResponse = chain.proceed(chain.request());  
                    return originalResponse.newBuilder()  
                        .body(new ProgressResponseBody(originalResponse.body(), progressListener))  
                        .build();  
            });  
        }  
    });  
  
    String[] localVarAuthNames = new String[] { };  
    return apiClient.buildCall(localVarPath, "PUT", localVarQueryParams, localVarCollectionQueryParams, localVarVa  
}
```

Generated Code (Java + Retrofit2)

```
public ApiClient(String authName, String clientId, String secret, String user
    this(authName);
    this.getTokenEndPoint()
        .setClientId(clientId)
        .setClientSecret(secret)
        .setUsername(username)
        .setPassword(password);
}

public void createDefaultAdapter() {
    json = new JSON();
    okBuilder = new OkHttpClient.Builder();

    String baseUrl = "http://localhost:8080";
    if (!baseUrl.endsWith("/"))
        baseUrl = baseUrl + "/";

    adapterBuilder = new Retrofit
        .Builder()
        .baseUrl(baseUrl)
        .addConverterFactory(ScalarsConverterFactory.create())
        .addConverterFactory(GsonConverterFactory.create(json.getGson()));
}

public <S> S createService(Class<S> serviceClass) {
    return adapterBuilder
        .client(okBuilder.build())
        .build()
        .create(serviceClass);
}
```

```
import retrofit2.Call;
import retrofit2.http.*;

import okhttp3.RequestBody;
import okhttp3.ResponseBody;

import io.swagger.client.model.ResponseDTO;
import io.swagger.client.model.TODO;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public interface TodoRestControllerApi {
    * @param id (required)
    @DELETE("api/todo/{id}")
    Call<TODO> deleteTodo4(
        @retrofit2.http.Path("id") Integer id
    );

    * @param id (required)
    @GET("api/todo/{id}")
    Call<TODO> get5(
        @retrofit2.http.Path("id") Integer id
    );

    * @return Call<List<TODO>>
    @GET("api/todo")
    Call<List<TODO>> list4();

    * @param body (required)
    @Headers({
        "Content-Type:application/json"
    })
    @POST("api/todo")
    Call<ResponseDTO> postTodo4(
        @retrofit2.http.Body TODO body
    );
}
```

Sample Retrofit2 Java Client

```
public void testSwaggerCliRetrofit2() {
    ApiClient apiClient = new ApiClient();
    TodoRestControllerApi svc = apiClient.createService(TodoRestControllerApi.class);

    Call<List<TodoDTO>> call = svc.list4();
    List<TodoDTO> res = executeGetBody(call, "GET /api/todo");

    System.out.println("http GET /api/todo => got " + res.size() + " elts");
}

private static <T> T executeGetBody(Call<T> call, String callDescr) {
    Response<T> resp;
    try {
        resp = call.execute();
    } catch (IOException ex) {
        throw new RuntimeException("can not call http " + callDescr, ex);
    }
    if (! resp.isSuccessful()) {
        throw new RuntimeException("Failure in http " + callDescr + ", " +
            "code:" + resp.code() + " message:" + resp.message());
    }
    return resp.body();
}
```