

Asynchronous communication models for JAX-RPC 2.0 (JSR-224)

Version 1.0

20 Nov 2003

Toshiyuki Kimura

R & D Headquarters
NTT DATA CORPORATION

Table of Contents

1. Introduction
2. Application level Modes of Interaction
 - 1) Synchronous Request-response Mode
 - 2) One-way RPC Mode
 - 3) Non-blocking RPC invocation
3. Application models
4. Types of asynchronous communication
5. Asynchronous communication models
 - 1) Synchronous transport with pull-based client APIs
 - 2) Synchronous transport with push-based client APIs
 - 3) Pull method via asynchronous transport with pull-based client APIs
 - 4) Pull method via asynchronous transport with push-based client APIs
 - 5) Push method via asynchronous transport with pull-based client APIs
 - 6) Push method via asynchronous transport with push-based client APIs
6. Issues
7. Policy direction
8. Basic design

1. Introduction

In the JAX-RPC 1.1 version (JSR-101), three kinds of modes (see section 2) are described as ‘Application level Modes of Interaction’. However, this version of JAX-RPC specification does not require the third one, ‘Non-blocking RPC invocation’. These are strongly related for the following two things:

- The JAX-RPC 1.1 specification requires supporting HTTP 1.1 as the fundamental transport for SOAP messages, and HTTP is a synchronous request-response protocol.
- The JAX-RPC 1.1 specification requires that a service client be able to participate in a session with a service endpoint, but it does not require session management as part of the interoperability requirements.

However, the JAX-RPC 2.0 version (JSR-224) is addressed to support client side asynchronous operations and session management mechanism. Now, we would like to propose a couple of asynchronous models and some related issues on this document.

2. Application level Modes of Interaction

The JAX-RPC 1.1 version (JSR-101) describes the following three ‘Application level Modes of Interaction’ as R012 at section ‘3 Requirements’.

1) Synchronous Request-response Mode

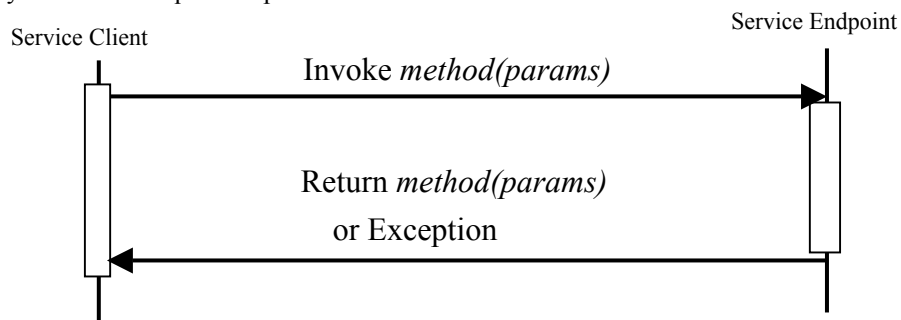


Fig. 1

2) One-way RPC Mode

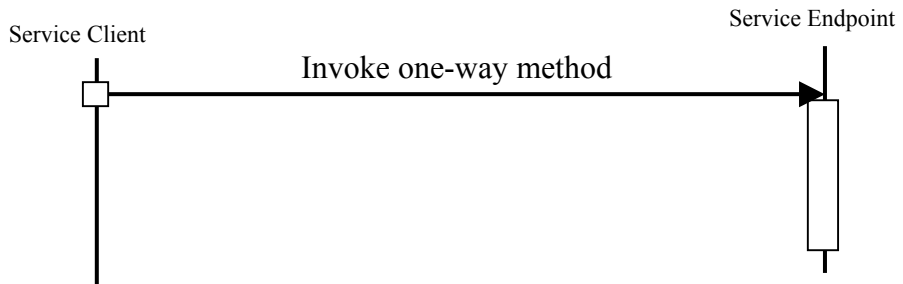


Fig. 2

3) Non-blocking RPC invocation

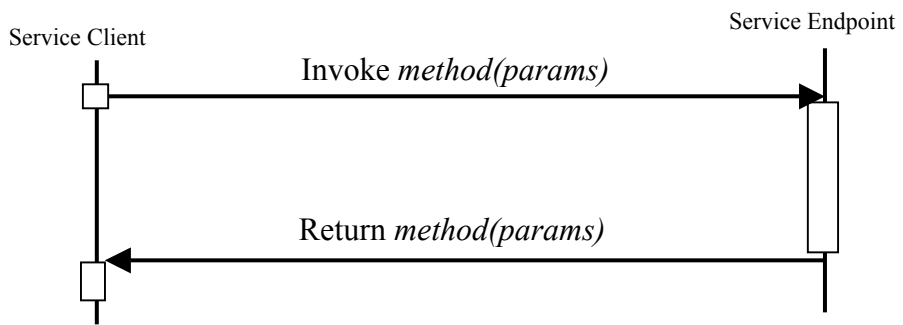


Fig. 3

3. Application models

The following diagram shows an application model on JAX-RPC for server-side and client-side. In the both side, all parts can be classified into two layers as ‘User application layer’ (in short, User AP) and ‘JAX-RPC runtime layer’ (in short, Runtime).

At this time, we have a plan to add an asynchronous feature for client side as a part of our design goal. And the current version of editors draft (October 17, 2003) says, “**Asynchrony** JAX-RPC 2.0 will add support for client side asynchronous operations”. This means we should specify the signatures and behaviors of an interface edge that is in the following diagram as ‘IF-1’. In addition, we’d better put into the scope of consideration also about ‘IF-2’ as communication-models between the client-side of JAX-RPC runtime and the server-side one.

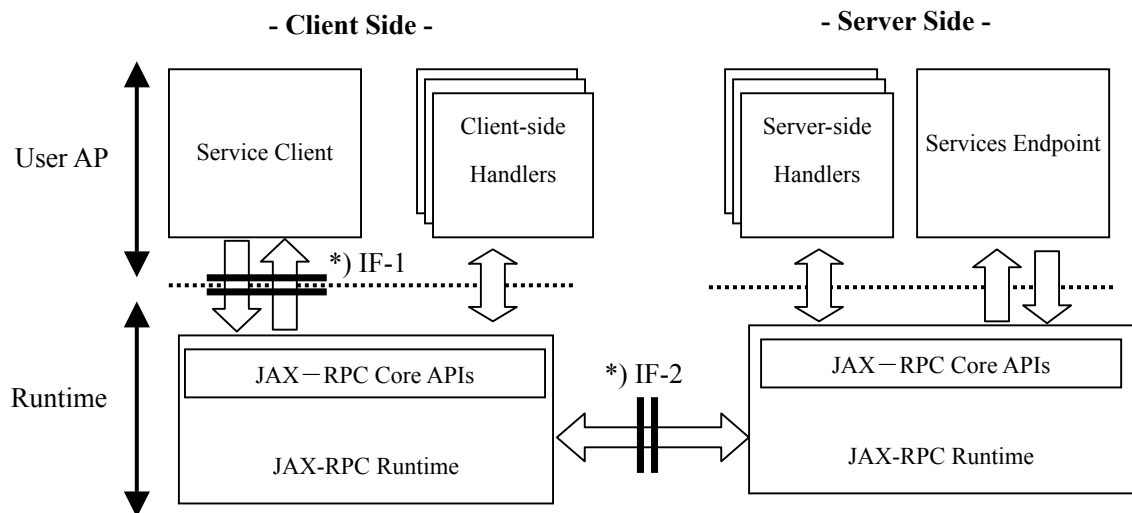


Fig. 4

4. Types of asynchronous communication

In general, there're two types of asynchronous communication mode. One is ‘Pull (Polling)’ type, and the other is ‘Push (Callback)’ type. This document is carrying out for both of types to make clear all of asynchronous communication models.

5. Asynchronous communication models

This chapter indicates a number of asynchronous models paying attention to three actors – ‘Service Client’, ‘Client-side JAX-RPC Runtime’ and ‘Service Endpoint’.

1) Synchronous transport with pull-based client APIs

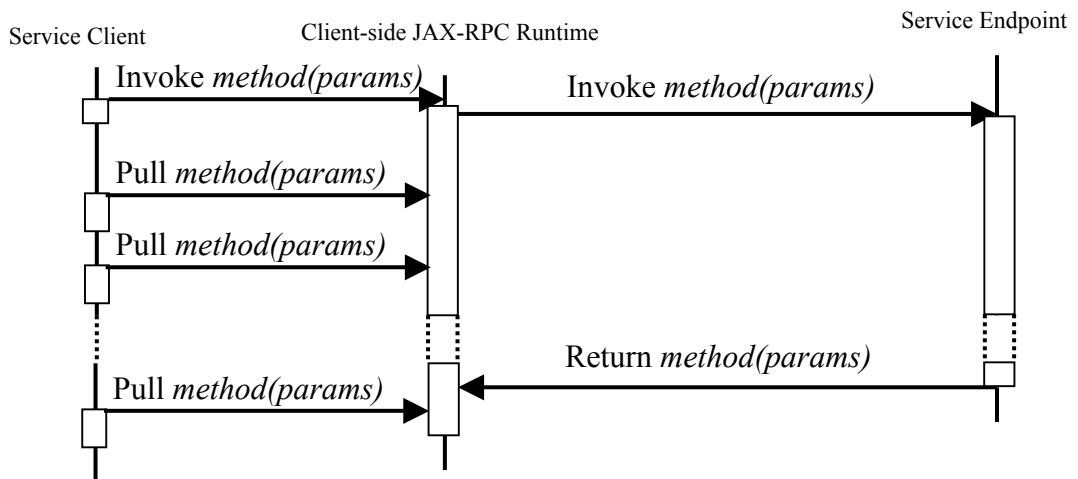


Fig. 5

2) Synchronous transport with push-based client APIs

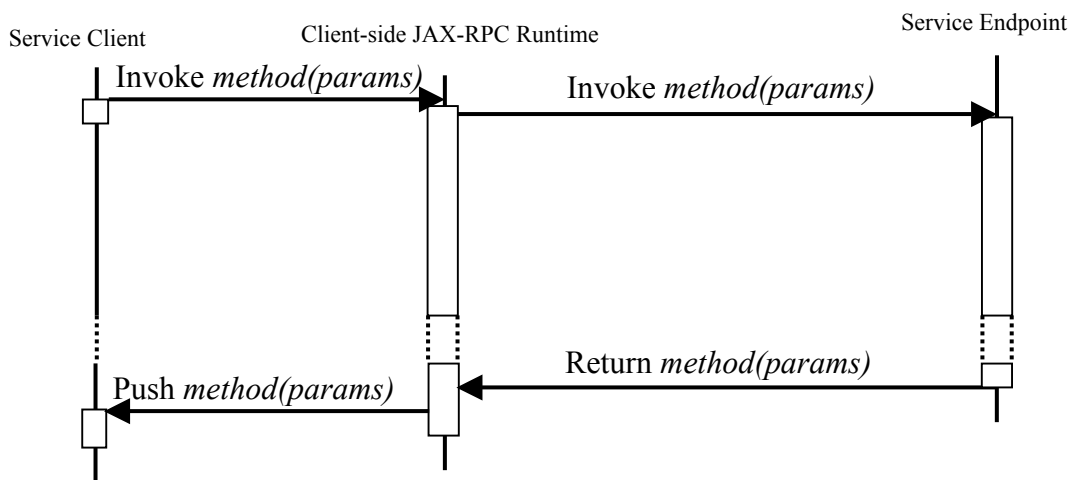


Fig. 6

3) Pull method via asynchronous transport with pull-based client APIs

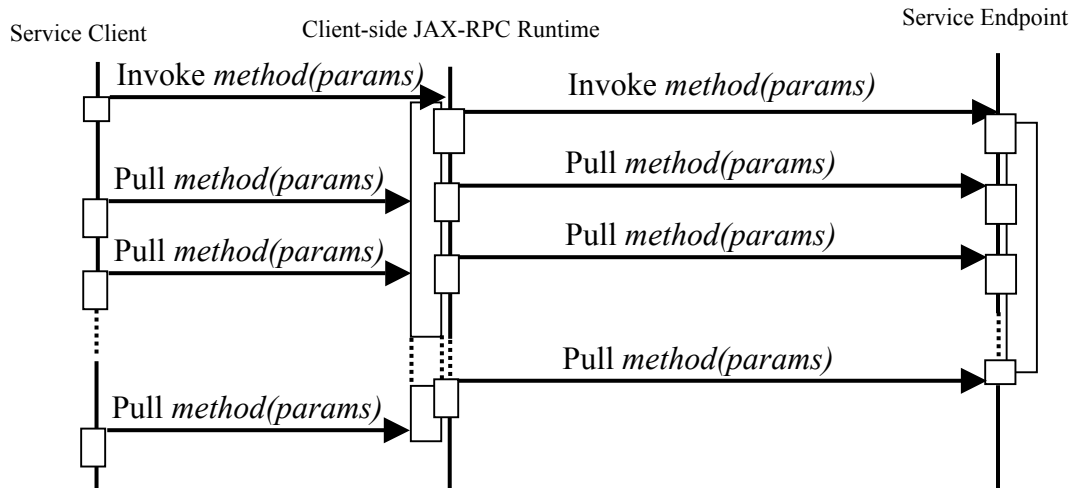


Fig. 7

4) Pull method via asynchronous transport with push-based client APIs

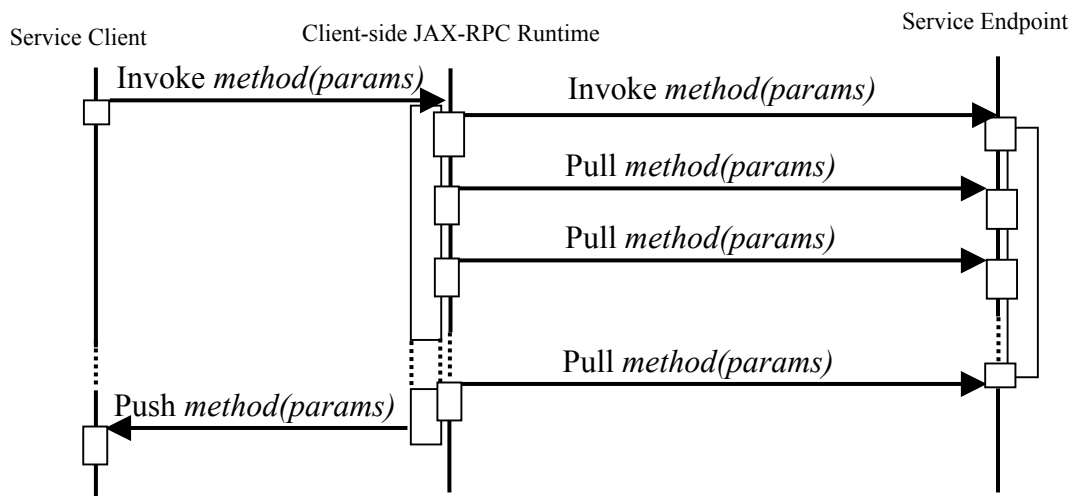


Fig. 8

5) Push method via asynchronous transport with pull-based client APIs

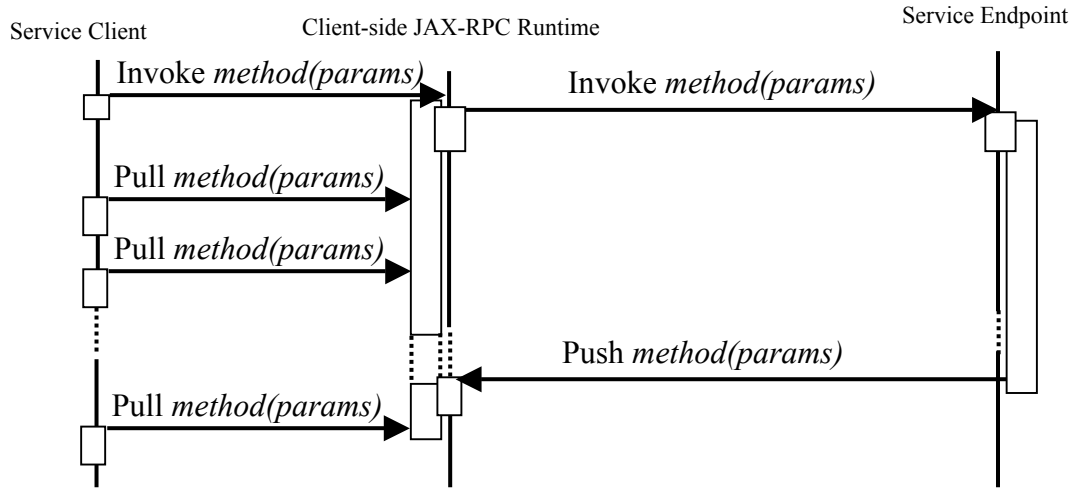


Fig. 9

6) Push method via asynchronous transport with push-based client APIs

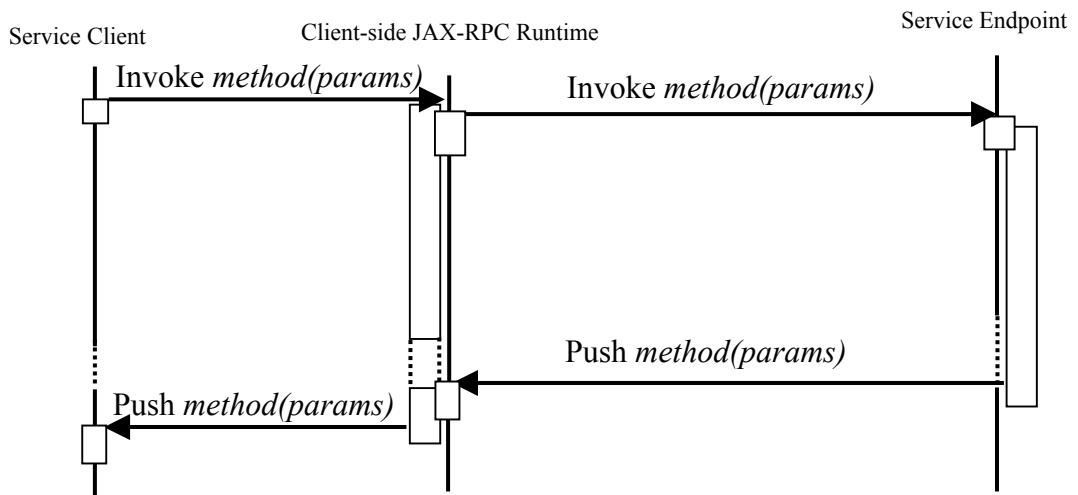


Fig. 10

6. Issues

This chapter describes some issues of each communication models that are described above.

ID	Types of IF	Client side API	Transport	Issues
0	Synchronous IF	Blocking	Request-Response	[1] [2]
1	Asynchronous IF	Polling	Request-Response	[2]
2		Callbacks	Request-Response	
3		Polling	Polling	[3] [4]
4		Callbacks	Polling	
5		Polling	Callbacks	[5]
6		Callbacks	Callbacks	

List of issues:

[1] Blocking behavior

Sometimes, a service client wants to continue processing in the same thread without waiting for the return of the remote method invocation. In this case, it's not welcomed that a JAX-RPC runtime blocks future processing.

[2] Wasting network resources

The request-response model is famous for HTTP protocol. A connection will be held until an answer is returned. It may become a problem as wasting of network resources on the server, when the residence time of the processing is comparatively long.

[3] Wasting server resources

The polling mechanism provides a flexible accessibility to a service client. However, it has a risk to waste server resources because of a heavy-duty polling from the service client.

[4] Service control initiative

In general, an initiative of service control should be in the service provider side (i.e. server side). But, the polling mechanism shifts the initiative to a service client side.

[5] Client programming model

Basically, a service client is a standalone console application. However, the bi-directional communication requires a SOAP server on the client side. It means that the callback method forces using new client programming model as follows:

- The service client has to have a SOAP server that is accessible from a target endpoint, even if the service client exists inside of their firewall.
- The service client has to have a deployment mechanism for a callback service.

7. Policy direction

This chapter describes our policy direction for above issues.

ID	Client side API	Transport	Proposed direction
1	Polling	Request-Response	Support
2	Callbacks	Request-Response	Support
3	Polling	Polling	Support
4	Callbacks	Polling	Support
5	Polling	Callbacks	Defer (*)
6	Callbacks	Callbacks	Defer (*)

NOTE (*): Definition of this interface is not so difficult. But, if we also turn the issues into an object of measures, we have to break into a deployment mechanism deeply.

8. Basic design

This chapter shows an illustrative architecture how we can implement asynchronous features that are described in the previous chapter. The hatching area is added.

