



JAX-RPC 仕様 (JSR-224) のための 非同期通信モデルの検討資料

第 1.0 版

2003 年 11 月 20 日

株式会社 NTT データ

技術開発本部

木村 利幸

目 次

1. はじめに
2. アプリケーションレベルの相互作用モデル
 - 1) 同期型要求／応答モード
 - 2) 一方向 RPC モード
 - 3) 非ブロック型のRPC呼び出し
3. アプリケーションモデル
4. 非同期通信モデルの種別
5. 非同期通信モデル
 - 1) 同期型トランスポートへのプル型 API 適用
 - 2) 同期型トランスポートへのプッシュ型API適用
 - 3) プル型トランスポートへのプル型API適用
 - 4) プル型トランスポートへのプッシュ型 API 適用
 - 5) プッシュ型トランスポートへのプル型 API 適用
 - 6) プッシュ型トランスポートへのプッシュ型 API 適用
6. 課題
7. 施策方針
8. 基本設計

1. はじめに

JAX-RPC バージョン 1.1 仕様 (JSR-101) の中では、「アプリケーションレベルの相互作用モデル」として、3つのモデル(2章参照)が定義されている。しかし、当該バージョンでは、第三のモデルである「非ブロック型のRPC呼び出し」を要件に加えてない。それは、以下の2つの事柄に深く関連している。

- JAX-RPC ver 1.1 仕様は、SOAP メッセージを転送するトランスポートとして HTTP 1.1 を基本としており、HTTP は要求/応答型の同期プロトコルである
- JAX-RPC ver 1.1 仕様は、セッション管理の実現を規定しているものの、それらの相互接続性については、規定していない

しかしながら、JAX-RPC ver 2.0 仕様 (JSR-224) は、「クライアント側の非同期性」および「セッション管理機構」の実現に取り組む方針である。ここで、本資料では非同期モデルと関連する課題について言及する。

2. アプリケーションレベルの相互作用モデル

JAX-RPC ver 1.1 仕様 (JSR-101) では、「第3章 要求条件」のR012として、3つのアプリケーションレベルの相互作用モデルを規定している。

1) 同期型要求/応答モード

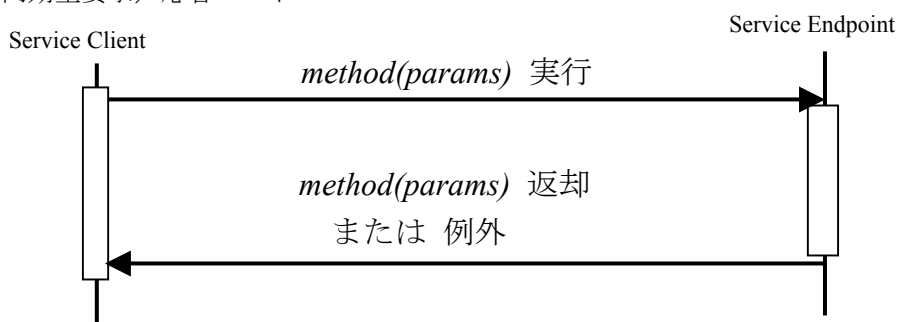


図. 1

2) 一方向RPCモード

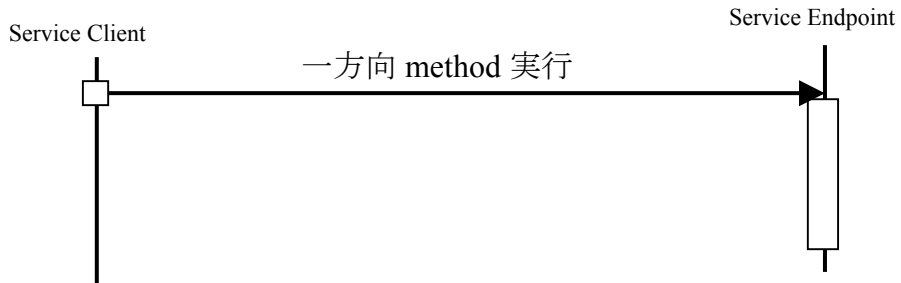


図. 2

3) 非ブロック型のRPC呼び出し

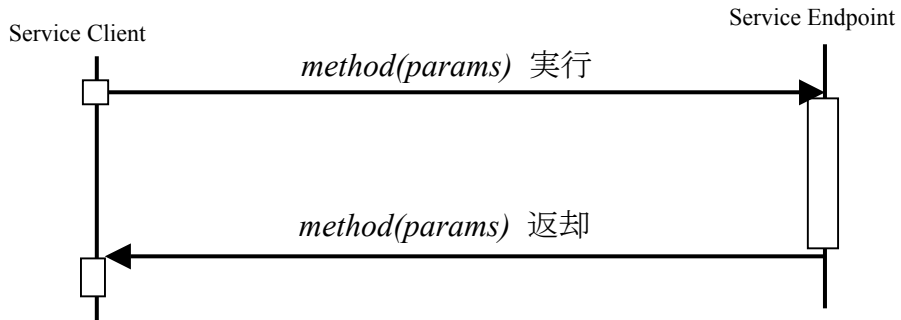


図. 3

3. アプリケーションモデル

下図は、JAX-RPCにおけるサーバ側およびクライアント側でのアプリケーションモデルを示している。双方において、全ての要素は、「ユーザアプリケーション層(以下、ユーザAP)」と「JAX-RPCランタイム層(以下、ランタイム)」の2層に分類される。

今回、私たちはクライアント側の非同期機能を盛り込むことを目標の一つに掲げている。そして、2003年10月17日時点での執筆者ドラフト版では、『JAX-PRC 2.0は、クライアント側の非同期呼び出しについて対応を予定している』と記述されている。これは、図4において「IF-1」として示されているインターフェース境界のシグネチャや挙動について定義する必要性があることを意味している。更に、クライアント・サーバのJAX-RPCランタイム間での通信モデルである「IF-2」の規定にも言及すべきであると思われる。

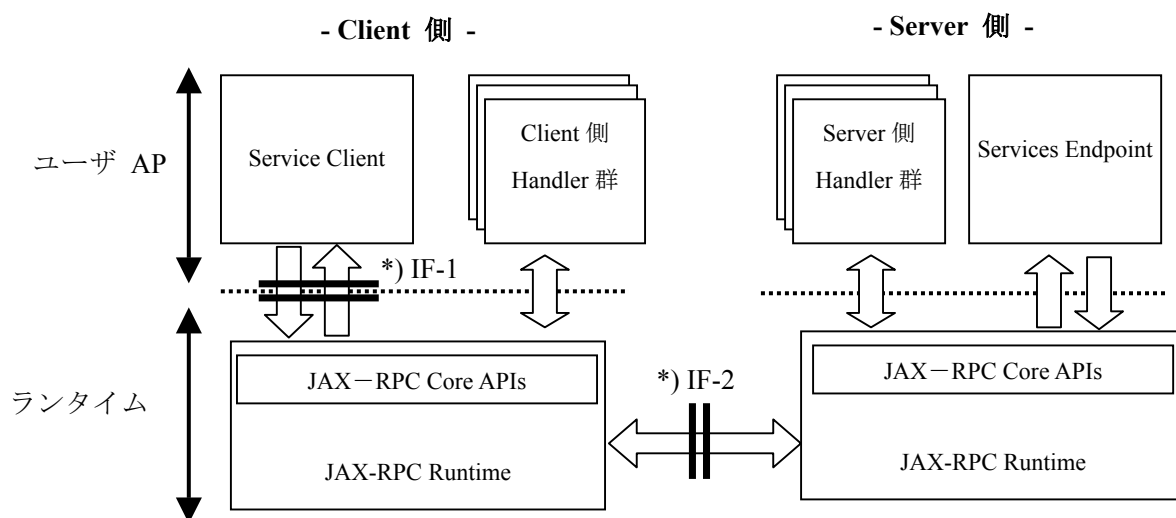


図. 4

4. 非同期通信モデルの種別

一般的に、2つの非同期通信モードが存在する。一つは、プル（ポーリング）型、もう一つは、プッシュ（コールバック）側である。本資料では、双方の型について触れ、全ての非同期通信モデルについて明確に規定する。

5. 非同期通信モデル

本章では、「サービスクライアント」「クライアント側JAX-RPCランタイム」「サービスエンドポイント」という3つのアクターに注目し、複数の非同期通信モデルについて説明する。

1) 同期型トランスポートへのプル型API適用

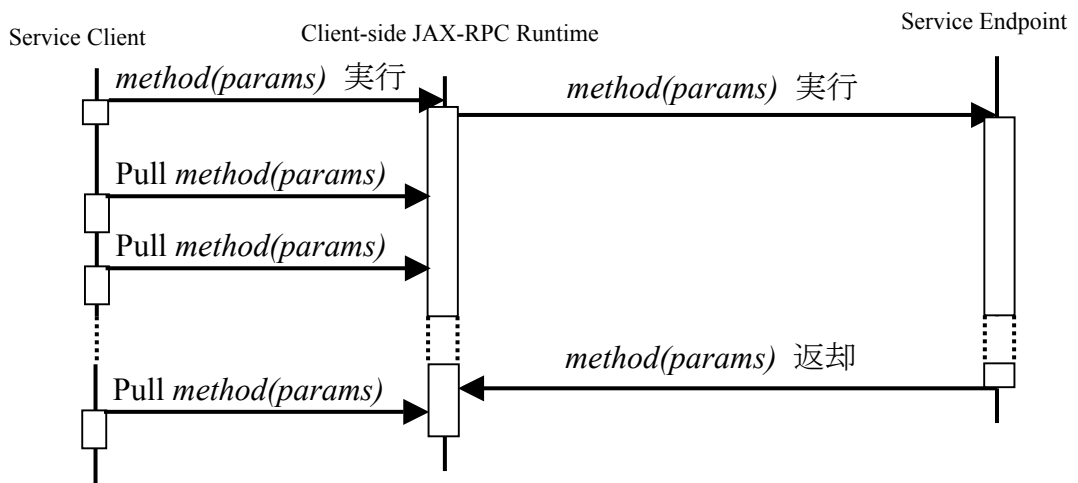


図. 5

2) 同期型トランスポートへのプッシュ型API適用

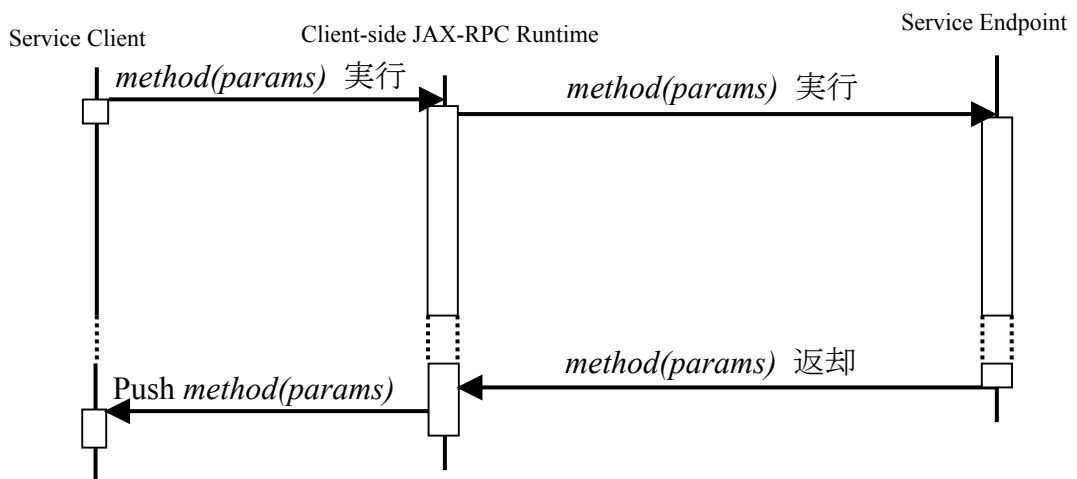


図. 6

3) プル型トランスポートへのプル型API適用

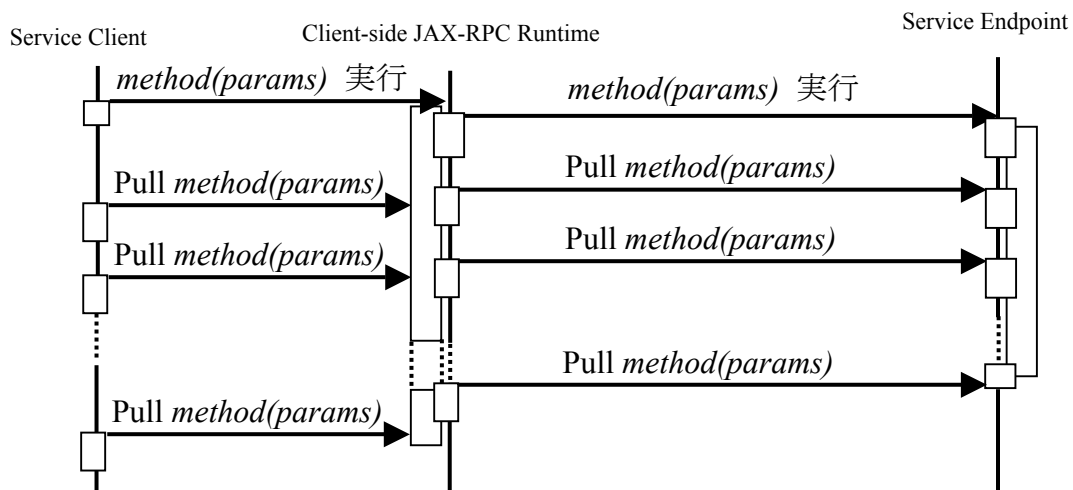


図. 7

4) プル型トランスポートへのプッシュ型API適用

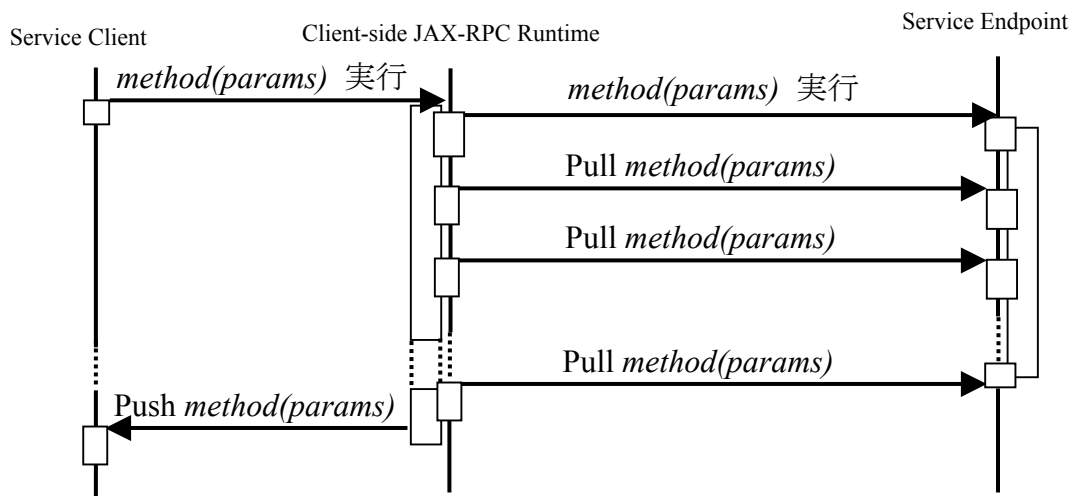


図. 8

5) プッシュ型トランスポートへのプル型API適用

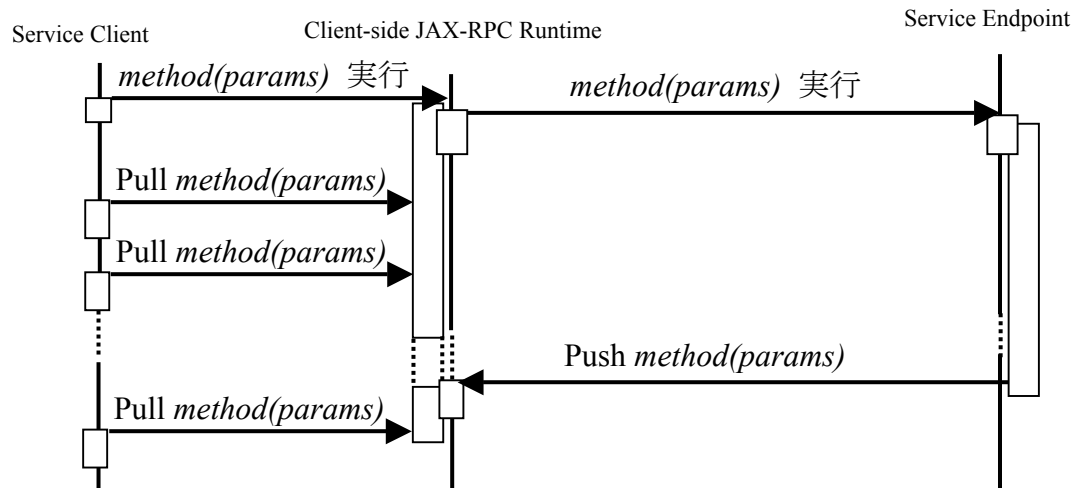


図. 9

6) プッシュ型トランスポートへのプッシュ型API適用

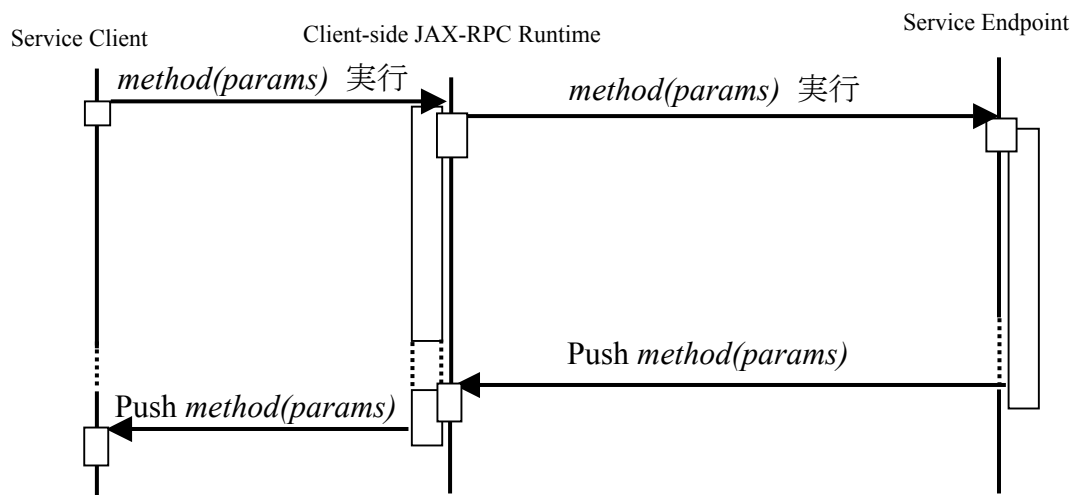


図. 10

6. 課題

この章では、前述の通信モデルそれぞれに関する課題について述べる。

ID	IF 型	クライアント側 API	トランスポート	課題番号
0	同期型 IF	ブロック型	要求応答(同期)型	[1] [2]
1	非同期型 IF	ポーリング型	要求応答(同期)型	[2]
2		コールバック型	要求応答(同期)型	
3		ポーリング型	ポーリング型	[3] [4]
4		コールバック型	ポーリング型	
5		ポーリング型	コールバック型	[5]
6		コールバック型	コールバック型	

課題一覧：

[1] 後続処理のブロック

クライアントは、Web サービスの呼び出し結果を待たず、そのまま同じスレッド内で処理を継続したいというケースがある。この場合、JAX-RPC ランタイム内での後続処理のブロックという挙動は、好ましくない。

[2] ネットワーク資源の浪費

要求応答型の通信モデルは、HTTP プロトコルで非常に有名である。サーバとの接続は、その応答が返されるまで保持されることになる。従って、サーバでの処理時間が非常に長い場合に、ネットワーク資源の浪費に繋がる可能性がある。

[3] サーバ資源の浪費

ポーリング機構は、クライアントに柔軟なアクセス性を提供する。しかし、クライアントからの過剰ポーリングによってサーバ資源を浪費する危険性を秘めている。

[4] サービス制御の主導権

一般的に、サービス制御の主導権は、サービス提供側であるサーバ側にあるべきである。しかし、ポーリング機構は、その主導権をクライアント側にシフトしてしまう。

[5] クライアント・プログラミング・モデル

基本的に、Web サービスクライアントは、スタンドアローンのコンソールアプリケーションである。しかし、双方向のコミュニケーションモデルは、クライアント側に SOAP サーバ機構を要求する。これは、コールバック方式は、次に示される新しいクライアント・プログラミング・モデルを要求するということを意味する。

- － クライアントがファイア・ウォール内に存在する場合であっても、Web サービスクライアントはエンドポイントからのコールバックを可能とするための SOAP サーバ機能を保有する必要がある。
- － クライアントは、コールバック用の Web サービスをクライアント側 SOAP サーバにデプロイするための機構を必要とする。

7. 施策方針

この章では、上記課題に対する対策の方針を示す。

ID	クライアント側 API	トランスポート	方針案
1	ポーリング型	要求応答(同期)型	対応
2	コールバック型	要求応答(同期)型	対応
3	ポーリング型	ポーリング型	対応
4	コールバック型	ポーリング型	対応
5	ポーリング型	Callbacks	先送り (*)
6	コールバック型	Callbacks	先送り (*)

脚注(*): このインターフェース定義は、さほど困難ではない。しかし、当該項目も対象とした場合は、デプロイ機構について深く踏み込む必要性が出てくる。

8. 基本設計

この章は、前述の非同期機能を如何に実現するかについてイラストを示している。下図で、網がけされている個所が追加されている。

