* --qo21Charlotte

**[Bayeux协议](http://www.blogjava.net/menlong999/archive/2009/08/16/291365.html)**

Posted on 2009-08-16 17:42 [Lv Yuanfang](http://www.blogjava.net/menlong999/) 阅读(2088) [评论(0)](http://www.blogjava.net/menlong999/archive/2009/08/16/291365.html#Post)  [编辑](http://www.blogjava.net/menlong999/admin/EditPosts.aspx?postid=291365)  [收藏](http://www.blogjava.net/menlong999/AddToFavorite.aspx?id=291365) http://www.blogjava.net/menlong999/aggbug/291365.html?webview=1

Bayeux协议  
（Bayeux真的不知道怎么翻。。。）

**Bayeux 协议-- Bayeux 1.0草案1**

**本备忘录状态**

This document specifies a protocol for the Internet community, and requests discussion and suggestions for improvement. This memo is written in the style and spirit of an IETF RFC but is not, as of yet, an official IETF RFC. Distribution of this memo is unlimited. This memo is written in UK English.

这份文档为互联网社区详细描述了一个协议，请求讨论并提出改进建议。此备忘录以 IETF RFC 风格写作，但是它不是、也还没有成为官方IETF RFC。此备忘录使用英式英文，对分发无限制。

**版权声明**

版权所有Dojo基金会 (2007). 保留所有权利

**摘要**

Bayeux is a protocol for transporting asynchronous messages (primarily over HTTP), with low latency between a web server.

Bayeux是在和web服务器间提供低时延、传输异步消息的协议（主要基于HTTP）。

**目录**

**1. 介绍**

**1.1. 目的**

**1.2. 需求**

**1.3. 术语**

**1.4. 总体运作**

**1.4.1. HTTP**

**1.4.2. 非 HTTP 传输**

**1.4.3. Javascript**

**1.4.4. 客户端至服务端事件分发**

**1.4.5. 服务端至客户端事件分发**

**1.4.5.i 轮询传输**

**1.4.5.ii 流传输**

**1.4.6. 双连接运作**

**1.4.7. 连接协商**

**1.4.8. 不连接运作**

**1.5 状态表**

**1.5.1 客户端状态**

**2. 协议涵义**

**2.1. 基本组件**

**2.2.通道**

**2.2.1 通道通配符**

**2.2.2 Meta通道**

**2.2.3 Service通道**

**2.3. 版本**

**2.4. 客户端ID**

**2.5 消息**

**3. 消息字段定义**

**3.1. channel**

**3.2. version**

**3.3. minimumVersion**

**3.4. supportedConnectionTypes**

**3.5. clientId**

**3.6. advice**

**3.6.1. reconnect advice**

**3.6.2. interval advice**

**3.6.3. multiple-clients advice**

**3.6.4. hosts advice**

**3.7. connectionType**

**3.8. id**

**3.9. timestamp**

**3.10. data**

**3.11. connectionId**

**3.12. successful**

**3.13. subscription**

**3.14. error**

**3.15. ext**

**3.16. json-comment-filtered**

**4. Meta 消息定义**

**4.1. 握手**

**4.1.1. 握手请求**

**4.1.2. 握手响应**

**4.2. 连接**

**4.2.1. 连接请求**

**4.2.2. 连接响应**

**4.4. 断开**

**4.4.1. 断开请求**

**4.4.2. 断开响应**

**4.5. 订阅**

**4.5.1. 订阅请求**

**4.5.2. 订阅响应**

**4.6. 取消订阅**

**4.6.1. 取消订阅请求**

**4.6.2. 取消订阅响应**

**5. 事件消息定义**

**5.1. 发布事件消息**

**5.1.1. 发布请求**

**5.1.2. 发布响应**

**5.2. 分发事件消息**

**6. 传输**

**6.1. long-polling**

**6.1.1 long-polling 请求消息**

**6.1.2 long-polling 响应消息**

**6.2. callback-polling**

**6.2.1 callback-polling 请求消息**

**6.2.2 callback-polling 响应消息**

**7. 安全**

**7.1.鉴权**

**7.2. Ajax 劫持**

**8. 多frame 运作**

**8.1 服务端多frame侦测**

**8.2 客户端多frame侦测**

**9. service 通道的请求响应运作**

**1. 介绍**

**1.1. 目的**

The primary purpose of Bayeux is to support responsive 2 way interactions between web clients using Ajax and the web server.

Bayexu协议的主要目的，是在使用Ajax的web客户端和web服务器之间，提供2种交互应答方式。

Bayeux is a protocol for transporting asynchronous messages (primarily over HTTP), with low latency between a web server and a web browser. The messages are routed via named channels and can be delivered: server to client, client to server and client to client (via the server). By default, publish subscribe routing semantics are applied to the channels, but other routing models are also supported.

Bayeux是在web服务器和web浏览器间提供低时延、传输异步消息的协议（主要基于HTTP）。消息通过命名通道被路由，并被分发：服务端至客户端，客户端至服务端和客户端至客户端（通过服务器中转）。默认时，发布订阅路由语义应用于通道，但是其他路由模型也被支持。

Delivery of asynchronous messages from the server to a web client is often described as "server-push". The combination of server push techniques with an Ajax web application has been called "comet". Cometd is a project by the Dojo Foundation to provide multiple implementation of the Bayeux project in several programming languages.

服务器至web客户端的异步消息分发常被称之为“server-push”。web应用中混合Ajax和server push的技术被称为“comet”。 Cometd是Dojo基金会的一个项目，它提供Bayeux项目在不同编程语言中的多个实现。

Bayeux seeks to reduce the complexity of developing Comet-driven applications by allowing implementers to more easily interoperate, solve common message distribution and routing problems, and provide mechanisms for incremental improvement and extension.

Bayeux致力于减少开发 Comet驱动应用的复杂性，允许实现者更容易的交互，解决通用消息分发和路由问题，同时也提供可持续改进和扩展的机制。

**1.2. 需求**

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119. An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements for the protocols it implements. An implementation that satisfies all the MUST or REQUIRED level and all the SHOULD level requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST level requirements but not all the SHOULD level requirements for its protocols is said to be "conditionally compliant."

这份文档中，关键词“必须”、“禁止”、“需要的”、“必须”、“必须不”、“应该”、“不应该”、“推荐的”、“可能”、“可选的”要按照 RFC2119中所描述的来理解。如果一个本协议的实现，没有满足一个或多个“必须”或“需要的”级别的需求，那它就是不兼容的。如果一个本协议的实现，满足了所有“必须”或“需要的”级别的需求和所有“应该”级别的需求，那它就是“完全兼容”；一个满足“必须”级别的需求但并非满足所有“应该”级别的需求的协议实现，就是“部分兼容”。

**1.3. 术语**

This specification uses a number of terms to refer to the roles played by participants in, and objects of, Bayeux communication:

本规范使用一系列术语来引用Bayeux通信中所参与的角色以及对象。

**client**

A program that initiates communications. A HTTP client is a client that initiates TCP/IP connections for the purpose of sending HTTP requests. A Bayeux client initiates the Bayeux message exchange and will typically execute within a HTTP client, but it is likely to have Bayeux clients that execute within HTTP servers. Implementations may distinguish between Bayeux clients running within a HTTP client and Bayeux clients running within the HTTP server. Specifically server-side Bayeux clients MAY be privileged clients with access to private information about other clients (e.g. client IDs) and subscriptions.

**客户端**

初始化通信的程序。初始化TCP/IP连接以发送HTTP请求的HTTP客户端就是一个客户端。一个Bayeux客户端初始化Bayeux消息交换，并且通常会在HTTP客户端中执行，但是也可能在HTTP服务端中执行。运行于HTTP客户端的Bayeux客户端和运行于HTTP服务端中的Bayxue客户端，实现会有区别。具体来说，服务器端的Bayeux客户端可能更优先的访问其他客户端的私有信息（比如它们的client ID）和订阅。

**服务端**

An application program that accepts communications from clients. A HTTP server accepts TCP/IP connections in order to service HTTP requests by sending back HTTP responses. A Bayeux server accepts and responds to the message exchanges initiated by a Bayeux client.

**服务端**  
接受来自客户端通信的应用程序。一个HTTP服务器接受TCP/IP连接，通过返回HTTP响应来服务HTTP请求。一个Bayeux服务端接受和响应由Bayeux客户端发起的消息交换。  
  
**request**  
An HTTP request message as defined by section 5 of RFC 2616   
**请求**  
RFC2616 第5章节所定义的HTTP请求消息  
  
**response**  
 A HTTP response message as defined by section 6 of RFC 2616   
**响应**  
RFC2616 第6章节所定义的HTTP响应消息  
  
**message**  
A message is a JSON object exchanged between client and server for the purposed of implementing the Bayeux protocol as defined by sections 3, 4 and 5.   
**消息**  
一个消息是客户端和服务端交互的一个JSON对象，目的就是为了实现3、4、5章节中所定义的Bayeux协议。  
  
**event**  
Application specific data that is sent over the Bayeux protocol   
**事件**  
通过Bayeux协议所发送的应用相关的数据  
  
**envelope**  
The transport specific message formate that wraps a standard Bayeux message.   
信封  
用来封装标准Bayeux消息所传输的特定消息格式

**1.4. 总体运作**

**1.4.1. HTTP**

The HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and optional body content over a connection with a server. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible entity-body content.

HTTP是个请求/响应协议。客户端发送请求到服务端是基于一个到服务端的连接，所使用格式为请求方法(get/post..)，URI，协议版本，紧接着是包含请求描述符的一个类似MIME的消息，客户端信息，和可选的内容体。服务端用一个状态行来响应，包括消息的协议版本和一个成功或错误码(500/400..)，紧接着是包含服务端信息的类似MIME的消息，实体元信息，和可能的实体内容体。

The server may not initiate a connection with a client nor send an unrequested response to the client, thus asynchronous events cannot be delivered from server to client unless a previously issued request exists. In order to allow two way asynchronous communication, Bayeux supports the use of multiple HTTP connections between a client and server, so that previously issued requests are available to transport server to client messages.

服务器不会初始化一个到客户端的连接，也不会发送一个未请求的响应到客户端，因此异步事件不能从服务端分发到客户端，除非一个之前相关的连接已存在。为允许两路异步通信，Bayeux支持一个客户端和服务端间多HTTP连接的使用，因此之前相关的请求可用来传输服务端至客户端的消息。

The recommendation of section 8.1.4 of RFC 2616 is that a single user client SHOULD NOT maintain more than 2 connection with any server, thus the Bayeux protocol MUST NOT require any more than two HTTP requests to be simultaneously handled by a server in order to handle all application (Bayeux based or otherwise) running within a client.

RFC2616中8.1.4章节所建议的是一个单独的客户端不应该维护和任何服务端多于2个以上的连接，因此，Bayeux协议禁止为了在一个客户端中运行处理所有应用（基于Bayeux或其他），而有2个以上HTTP请求来被一个服务端同时处理(就是说只能有一个信道(一个客户端只能订阅一个信道，而不能因子自身的需求与服务端建立两个以上的信道)一个意思)，。

**1.4.2. 非HTTP传输**

While HTTP is the predominant transport protocol used on the internet, it is not intended that it will be the only transport for Bayeux. Other transports that support a request/response paradigm may be used. However this document assumes HTTP for reasons of clarity. When non-HTTP connection-level transport mechanisms are employed, conforming Bayeux servers and clients MUST still conform to the semantics of the JSON messages outlined in this document.

HTTP是在因特网上使用的占据统治地位的传输协议，并非是传输Bayeux的唯一方式。其他支持请求/响应模式的传输方式也可以被使用。为了清晰起见，本文假设使用HTTP。使用非HTTP连接级别传输机制时，符合Bayeux的服务端和客户端仍必须符合文本所描述的JSON消息语义。

Several of the "transport types" described in this document are distinguished primarily by how they wrap messages for delivery over HTTP and the sequence and content of the HTTP connections initiated by clients. While this may seem like a set of implementation concerns to observant readers, the difficulties of creating interoperable implementations without specifying these semantics fully is a primary motivation for the development of this specification. Were the deployed universe of servers and clients more flexible, it may not have been necessary to develop Bayeux.

本文所描述的多种传输方式，其区分主要在于如何包装在HTTP上投递的消息，顺序和客户端发起连接的HTTP内容。虽然这可能看起来像是一系列深层次读者才关切的实现，但创建可交互实现而不完全指定语义的难度是开发此规范的一个主要动机。要是世界范围的服务器和客户端能更灵活的部署，就没有必要开发Bayeux了。

Regardless, care has been taken in the development of this specification to ensure that future clients and servers which implement differing connection-level strategies and encodings may still evolve and continue to be conforming Bayeux implementations so long as they implement the JSON-based public/subscribe semantics outlined herein.

TODO？

The rest of this document speaks as though HTTP will be used for message transport.

本文剩余部分会使用HTTP来讲述消息传输。

**1.4.3. Javascript**

The majority of Bayeux clients will be implemented in JavaScript and will be running within the security framework of a client browser. For applications that need to communicate with multiple servers, the client implementation MUST adhere to the single origin policy for security.

多数Bayeux客户端会用JavaScript实现，并运行于一个客户端浏览器的安全框架内。为满足和多服务端的通信需要，基于安全考虑，客户端实现必须固守单源策略。

**1.4.4. 客户端至服务端的时间传送Client to Server event delivery**

A Bayeux event is sent from the client to the server via a HTTP request initiated by a user agent and transmitted to an origin server via a chain of zero or more intermediaries (proxy, gateway or tunnel):

一个Bayeux事件从客户端发送至服务端，是由一个用户代理发起的HTTP请求，经过若干环节的中介（代理，网关，或通道）被传送到原始服务器端。

BC ---------- U ---------- P ------------ O ---------- BS  
| --M0(E)--> | | | |  
| | ---HTTP request(M0(E))--> | |  
| | | | --M0(E)--> |  
| | | | <---M1---- |  
| | <---HTTP response(M1)---- | |  
| <---M1--- | | | |  
| | | | |

The figure above represents a Bayeux event E encapsulated in a Bayeux message M0 being sent from a Bayeux client BC to a Bayeux server BS via a HTTP request transmitted from a User Agent U to to an Origin server O via a proxy P. The HTTP response contains another Bayeux message M1 that will at least contain the protocol response to M0, but may contain other Bayeux events initiated on the server or on other clients.

如上图所示，Bayeux事件E被封装成Bayeux消息M0，从Bayeux客户端BC发往Bayeux服务端BS，是通过从用户代理U经由代理P到原始服务器O的HTTP请求传输的。HTTP响应所包含的另一个Bayeux消息M1，不仅包含M0的协议响应，也可能包含服务端或其他客户端发起的其他Bayeux事件。

**1.4.5. 服务端至客户端的事件传送Server to Client event delivery**

A Bayeux event is sent from the server to the client via a HTTP response to a HTTP request sent in anticipation by a user agent and transmitted to an origin server via a chain of zero or more intermediaries (proxy, gateway or tunnel):

一个Bayeux事件从服务端发送至客户端， 是由一个用户代理预先发起HTTP请求的HTTP响应，经过若干环节的中介（代理，网关，或通道），被传送到原始服务器端。

BC ---------- U ---------- P ------------ O ---------- BS  
| ---M0---> | | | |  
| | --- HTTP request(M0) ---> | |  
| | | | ----M0---> |  
~ ~ ~ ~ ~ wait  
| | | | <--M1(E)-- |  
| | <--HTTP response(M1(E))-- | |  
| <--M1(E)-- | | | |  
~ ~ ~ ~ ~

The figure above represents a Bayeux message M0 being sent from a Bayeux client BC to a Bayeux server BS via a HTTP request transmitted from a User Agent U to to an Origin server O via a proxy P. The message M0 is sent in anticipation of a Bayeux event to be delivered from server to client and the Bayeux server waits for such an event before sending a response. A Bayeux event E is shown being delivered via Bayeux message M1 in the HTTP response. M1 may contain zero, one or more Bayeux events destined for the Bayeux client.

如上图所示，来自Bayeux客户端BC的Bayeux消息M0，到达Bayeux服务端BS，是通过从用户代理U经由代理P到原始服务器O的HTTP请求传输的。消息M0被发送，并期望一个Bayeux事件要从服务端被传送到客户端，Bayeux服务端在发送响应前会等待这样的事件。Bayeux事件E在HTTP响应中由Bayeux消息M1发送。M1可能包含0个，1个或多个到Bayeux客户端的Bayeux事件。

The transport used may terminate the HTTP response after delivery of M1 or use techniques to leave the response open and stream additional messages to the client.

所使用的传输方式在M1发送后可能中断HTTP响应，或者使用技术保持响应打开并传递到客户端的其他消息。

**1.4.5.i 轮询传输Polling transports**

Polling transports will always terminate the HTTP response after sending all available Bayeux messages.

轮询传输方式总会在发送所有可用Bayeux消息后中断HTTP响应。

BC ---------- U ---------- P ------------ O ---------- BS  
| ---M0---> | | | |  
| | --- HTTP request(M0) ---> | |  
| | | | ----M0---> |  
~ ~ ~ ~ ~ wait  
| | | | <--M1(E)-- |  
| | <--HTTP response(M1(E))-- | |  
| <--M1(E)-- | | | |  
| ---M2---> | | | |  
| | --- HTTP request(M2) ---> | |  
| | | | ----M2---> |  
~ ~ ~ ~ ~ wait

On receipt of the HTTP response containing M1, the Bayeux client issues a new Bayeux message M2 either immediately or after an interval in anticipation of more events to be delivered from server to client. Bayeux implementations are required to support a specific style of polling transport called "long polling" (see sec 6.1).

当收到包含M1的HTTP响应时，Bayeux客户端立即发出一个新的Bayeux消息，或者在一个从服务端到客户端被发送的更多事件的期望间隔后发出。Bayeux实现要求支持一个被称为“长轮询”的特定风格的轮询传输。

**1.4.5.ii 流传输Streaming transports**

Some Bayeux transports use a streaming technique (also called a forever response) that allows multiple messages to be sent over the same HTTP response:

有些Bayeux传输方式使用流技术（也称为永久响应），流技术允许多个消息通过同一个HTTP响应发送。

BC ---------- U ---------- P ------------ O ---------- BS  
| ---M0---> | | | |  
| | --- HTTP request(M0) ---> | |  
| | | | ----M0---> |  
~ ~ ~ ~ ~ wait  
| | | | <--M1(E0)- |  
| | <--HTTP response(M1(E0))- | |  
| <--M1(E0)- | | | |  
~ ~ ~ ~ ~ wait  
| | | | <--M1(E1)- |  
| | <----(M1(E1))------------ | |  
| <--M1(E1)- | | | |  
~ ~ ~ ~ ~ wait

Streaming techniques avoid the latency and extra messaging of anticipatory requests, but are subject to the implementation of user agents and proxies as they requires incomplete HTTP responses to be delivered to the Bayeux client.

流技术避免延时和额外的信息预先请求，但是对用户代理和代理的实现有影响，因为他们需要不完整的HTTP响应以发送到Bayeux客户端。

**1.4.6. 双连接运作Two connection operation**

In order to achieve bi-directional communications, a Bayeux client will use two HTTP connections to a Bayeux server so that both server to client and client to server messaging may occur asynchronously:

为了实现双向通信，Bayeux客户端会使用两个HTTP连接连接到Bayeux服务端，因此服务端到客户端和客户端到服务端的消息传递可能异步产生:

BC ---------- U ---------- P ------------ O ---------- BS  
| ---M0---> | | | |  
| | ------ req0(M0) --------> | |  
| | | | ----M0---> |  
~ ~ ~ ~ ~ wait  
| --M1(E1)-> | | | |  
| | ----- req1(M1(E1))------> | |  
| | | | --M1(E1)-> |  
| | | | <---M2---- |  
| | <---- resp1(M2)---------- | |  
| <---M2--- | | | |  
~ ~ ~ ~ ~ wait  
| | | | <-M3(E2)-- |  
| | <-----resp2(M3(E2))------ | |  
| <-M3(E2)-- | | | |  
| ---M4---> | | | |  
| | ------req3(M4)----------> | |  
| | | | ----M4---> |  
~ ~ ~ ~ ~ wait

HTTP requests req0 and req1 are sent on different TCP/IP connections, so that the response to req1 may be sent before the response to req0. Implementations MUST control HTTP pipelining so that req1 does not get queued behind req0 and thus enforce an ordering of responses.

HTTP请求req0和req1在不同的TCP/IP连接上发送，因此req1的响应可能在req0的响应前被发送。实现必须控制HTTP流水线处理，因此req1不能在req0前被排队，因而能强制响应的顺序。

**1.4.7. 连接协商Connection Negotiation**

Bayeux connections are negotiated between client and server with handshake messages that allow the connection type, authentication and other parameters to be agreed upon between the client and the server.

Bayeux连接通过握手消息在服务端和客户端之间协商允许的连接类型，认证和其他在客户端和服务端约定的参数。

BC ----------------------------------------- BS  
| ------------------ handshake request ---> |  
| <---- handshake response ---------------- |  
| -------------------- connect request ---> |  
~ ~ wait  
| <------ connect response ---------------- |

Connection negotiation may be iterative and several handshake messages may be exchanged before a successful connection is obtained. Servers may also request connection renegotiation by sending an unsuccessful connect response with advice to reconnect with a handshake message.

获得一个成功的连接前，连接协商可能反复交换多个握手消息。服务端也可能要求连接重新协商，这要发送一个不成功的连接响应，并在握手消息中建议重连。

BC ----------------------------------------- BS  
| ------------------ handshake request ---> |  
| <-- unsuccessful handshake response ----- |  
| ------------------ handshake request ---> |  
| <-- successful handshake response ------- |  
| -------------------- connect request ---> |  
~ ~ wait  
| <------ connect response ---------------- |  
| -------------------- connect request ---> |  
| <---- unsucessful connect response ------ |  
| ------------------ handshake request ---> |  
| <-- successful handshake response ------- |  
| -------------------- connect request ---> |  
~ ~ wait  
| <------ connect response ---------------- |

**1.4.8. 未连接运作Unconnected operation**

OPTIONALLY, messages can be sent without a prior handshake (see 5.1 Publish event messages).

可选的，没有预先的握手，消息也能被发送。

| ------------------- message request ----> |  
| <---- message response ------------------ |

This pattern is often useful when implementing non-browser clients for Bayeux servers. These clients often simply wish to address messages to other clients which the Bayeux server may be servicing, but do not wish to listen for events themselves.

这种方式为Bayeux服务端实现非浏览器的客户端时通常会有用。这些客户端通常仅仅想发送消息到其他的有Bayeux服务端服务的客户端，而不想注意自身的事件。(只想着发过去，而不会考虑要不要保持握手从而更多地交换数据)

**1.5 状态表State Tables**

**1.5.1 客户端状态Client State**

-------------++------------+-------------+----------- +------------  
State/Event || handshake | Timeout | Successful | Disconnect  
|| request | | connect | request  
|| sent | | response | sent   
-------------++------------+-------------+----------- +------------  
UNCONNECTED || CONNECTING | UNCONNECTED | |  
CONNECTING || | UNCONNECTED | CONNECTED | UNCONNECTED  
CONNECTED || | UNCONNECTED | | UNCONNECTED  
-------------++------------+-------------+------------+------------

**2. 协议涵义Protocol values**

**2.1. 基本组件Common Elements**

The characters used for Bayeux names and identifiers are defined by the BNF definitions:

用作Bayeux名称和标识符的字符用BNF定义：

alpha = lowalpha | upalpha  
  
lowalpha = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |  
"j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |  
"s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"  
  
upalpha = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |  
"J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |  
"S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"  
  
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |  
"8" | "9"  
  
alphanum = alpha | digit  
  
mark = "-" | "\_" | "!" | "~" | "(" | ")" | "$" | "@"   
  
string = \*( alphanum | mark | " " | "/" | "\*" | "." )  
  
token = ( alphanum | mark ) \*( alphanum | mark )  
  
integer = digit \*( digit )

**2.2. 通道Channel**

Channels are identified by names that are styled as the absolute path component of a URI without parameters as defined by RFC2396.

TODO?

channel\_name = "/" channel\_segments  
channel\_segments = channel\_segment \*( "/" channel\_segment )  
channel\_segment = token

The channel name consists of an initial "/" followed by an optional sequence of path segments separated by a single slash "/" character. Within a path segment, the character "/" is reserved.

Channel names commencing with "/meta/" are reserved protocol use. Example non-meta channel names are:

通道名称以“/”开头，剩下为“/”分割的可选的路径段序列。一个路径段内，字符“/”被保留。“/meta/”开头的通道名称保留为协议使用。非元通道名称示例如下：

/foo/foo/bar/foo-bar/(foobar)

**2.2.1 通道通配符Channel Globbing**

A set of channels may be specified with a channel globbing pattern:

channel\_pattern = \*( "/" channel\_segment ) "/" wild\_card  
wild\_card = "\*" | "\*\*"

The channel patterns support only trailing wildcards of either "\*" to match a single segment or "\*\*" to match multiple segments. Example channel patterns are:

/foo/\* Matches /foo/bar and /foo/boo. Does not match /foo, /foobar or /foo/bar/boo. /foo/\*\* Matches /foo/bar, /foo/boo and /foo/bar/boo. Does not match /foo, /foobar or /foobar/boo  
一组通道可以用通道通配符指定：  
channel\_pattern = \*( "/" channel\_segment ) "/" wild\_card  
wild\_card = "\*" | "\*\*"  
通道模式仅支持“\*”支持单个段或“\*\*”支持多个段的末尾通配符。示例为：  
/foo/\* Matches /foo/bar and /foo/boo. Does not match /foo, /foobar or /foo/bar/boo. /foo/\*\* Matches /foo/bar, /foo/boo and /foo/bar/boo. Does not match /foo, /foobar or /foobar/boo

**2.2.2 Meta通道Meta Channel**

The channels within the "/meta/" segment are the channels used by the Bayeux protocol itself. Local server-side Bayeux clients MAY, and remote Bayeux clients SHOULD NOT, subscribe to meta channels. Messages published to meta channels MUST NOT be distributed to remote clients by Bayeux servers. A server side handler of a meta channel MAY publish response messages that are delivered only to the client that sent the original request message. If a message published to a meta channel contains an id field, then any response messages delivered to the client MUST contain an id field with the same value.

包含“/meta/”段的通道被Bayeux协议自身使用。本地服务器端Bayeux客户端可以，而远端Bayeux客户端不应该订阅meta通道。发布到meta通道的消息不可由Bayeux服务端分发到远端客户端。一个meta通道的服务端处理者可以发布消息，这些消息只能被发送到发送原始请求消息的客户端。如果发布到meta通道的消息包含id字段，发送到这个客户端的任何响应消息必须包含id字段，并且值相同。

**2.2.3 Service通道Service Channel**

The channels within the "/service/" channel segement are special channels designed to assist request/response style messaging. Messages published to service channels are not distributed to any remote Bayeux clients. Handlers of service channels MAY deliver response messages to the client that published the request message. Servers SHOULD NOT record any subscriptions they receive for service channels. If a message published to a meta channel contains an id field, then any response messages SHOULD contain an id field with the same value or a value derived from the request id. Request response operations are described in detail in section 9.

包含“/service/”段的通道是特殊的通道，被设计用来协助请求/响应风格的消息传送。发布到service通道的消息不被分发到任何远端Bayeux客户端。service通道的处理者可以发送响应消息给发布请求消息的客户端。服务端不应该记录任何所接收到的对service通道的订阅。如果发布到meta通道的消息包含id字段，任何响应消息应该也包含id字段，并且值相同或者值为从请求id的派生值。请求响应操作在第9章节详细描述。

**2.3. 版本Version**

A protocol version is a integer followed by an optional "." separated sequence of alphanumeric elements:

version = integer \*( "." version\_element )  
version\_element = alphanum \*( alphanum | "-" | "\_" )

Versions are compared element by element, applying normal alphanumeric comparison to each element.

协议版本是一个整数，后面是一个可选的“.”分割的字母数字部分序列：

version = integer \*( "." version\_element )  
version\_element = alphanum \*( alphanum | "-" | "\_" )

版本可以逐部分来比较，每部分使用通常的字母数字比较方式。

**2.4. 客户端IDClient ID**

A client ID is an random, non predictable sequence of alpha numeric characters:

clientId = alphanum \*( alphanum )

Client IDs are generated by the server and SHOULD be created with a strong random algorithm that contains at least 128 truly random bits. Servers MUST ensure that client IDs are unique and SHOULD attempt to avoid reuse of client IDs. Client IDs are encoded for delivery as JSON strings.

客户端ID是一个随机数，不可预计的字母数字字符序列：

clientId = alphanum \*( alphanum )

客户端ID由服务端生成，应该用强大的随机数算法创建，至少要包含128位精确的随机位。服务端必须确保客户端ID唯一并应该尽量重用客户端ID。客户端ID被编码成JSON字符串传送。

**2.5 消息Messages**

Bayeux messages are JSON encoded objects that contain an unordered sequence of name value pairs representing fields and values. Values may be a simple strings, numbers, boolean values, or complex JSON encoded objects. A Bayeux message MUST contain one and only one channel field which determines the type of the message and the allowable fields.

All Bayeux messages SHOULD be encapsulated in a JSON array so that multiple messages may be transported together. A Bayeux client or server MUST accept either array of messages and MAY accept a single message. The JSON message or array of messages is itself often encapsulated in transport specific formatting and encodings. Below is an example Bayeux message in a JSON array representing an event sent from a client to a server:

[  
{  
"channel": "/some/name",  
"clientId": "83js73jsh29sjd92",  
"data": { "myapp" : "specific data", value: 100 }  
}  
]  
Bayeux消息用JSON对象编码，JSON对象包含表示字段和值的没有顺序的明值对序列。值可以是简单字符串，数字，布尔值，或者复杂的JSON编码的对象。一个Bayeux消息必须有且仅有一个channel字段，它决定了消息类型和可以容许的字段。  
所有的Bayeux应该封装到一个JSON数组中，因此多个消息可以一起传输。一个Bayeux客户端或服务端必须接受消息数组，也可以接受单个消息。JSON消息或消息数组自身常被封装成传输方式特定的格式和编码。下面是一个Bayeux消息的例子，使用JSON数组表示从一个客户端到一个服务端发送的事件。

**3. 消息字段定义Message Field Definitions**

**3.1. 通道channel**

The channel message field MUST be included in every Bayeux message to specify the source or destination of the message. In a request, the channel specifies the destination of the message, and in a response it specifies the source of the message.

通道消息字段必须包含在每个Bayeux消息中，以指定消息的源或目的。在一个请求中，通道指定了消息的目的地，而在一个响应中它制定了消息的源。

**3.2. 版本version**

The version message field MUST be included in messages to/from the "/meta/handshake" channel to indicate the protocol version expected by the client/server.

版本消息字段必须包含在到或者来自“/meta/handshake”通道的消息中，表明客户端/服务端期望的协议版本。

**3.3. 最小版本minimumVersion**

The minimumVersion message field MAY be included in messages to/from the "/meta/handshake" channel to indicate the oldest protocol version that can be handled by the client/server.

最小版本消息字段可以包含在到或者来自“/meta/handshake”通道的消息中，表明客户端/服务端所能处理的最低协议版本。

**3.4. 支持连接类型supportedConnectionTypes**

The supportedConnectionTypes field is included in messages to/from the "/meta/handshake" channel to allow clients and servers to reveal the transports that are supported. The value is an array of strings, with each string representing a transport name. Defined connection types include:

支持连接类型字段包含在到或者来自“/meta/handshake”通道的消息中，允许客户端和服务端暴露支持的传输方式。值为字符串数组，每个字符串代表一种传输方式名称。定义的连接类型包括：

**long-polling**  
messages are sent to the server as the 'message' parameter of a application/x-www-form-urlencoded encoded POST request. Messages are sent to the client as unencapsulated body content of a POST response. This transport is defined in section [XXX] of this memo.   
**长轮询**  
消息通过application/x-www-form-urlencoded编码的POST请求的message参数发送到服务端。消息通过未封装的POST响应体发送客户端。这种传输方式定义在本备忘录的XXX章节中。  
  
**callback-polling**  
messages are sent to the server as the 'message' parameter of a url encoded GET request. Responses are sent wrapped in a JavaScript callback in order to facilitate delivery. As specified by the JSON-P pseudo-protocol, the name of the callback to be triggered is passed to the server via the 'jsonp' GET parameter. In the absence of such a parameter, the name of the callback defaults to 'jsonpcallback'.   
**回调轮询**  
消息通过通过url编码的GET请求的message参数发送到服务端。为容易发送，响应被封装到JavaScript回调函数中发送。按照JSON-P伪协议规范，被触发的回调函数名称作为jsonp GET参数传送到服务端。如果此参数不存在，回调函数名字默认为jsonpcallback。  
  
**iframe**  
OPTIONAL transport using the document content of a hidden iframe element.   
**iframe**  
可选的 用一个隐藏iframe元素的文档内容传输  
  
**flash**  
OPTIONAL transport using the capabilities of a browser flash plugin.

**flash**

可选的 用浏览器的flash插件功能传输

All server and client implementations MUST support the "long-polling" connection type and SHOULD support "callback-polling". All other connection types are OPTIONAL.

所有服务端和客户端实现必须支持“长轮询”连接类型，并且应该支持“回调轮询”。其他连接类型都是可选的。

**3.5. clientId**

The clientId message field uniquely identifies a client to a Bayeux server. The clientId message field MUST be included in every message sent to the server except for a messages sent to the "/meta/handshake" channel and a publish message (see 5.1 Publish event messages). The clientId field MUST be returned in every message response except for a failed handshake request and is OPTIONAL in a message delivery message.

clientId消息字段唯一标识到一个Bayeux服务端的一个客户端。clientId消息字段必须包含在每个发往服务端的哦消息中，除了发往/meta/channel通道的消息和publish消息（参考 5.1 发布事件消息）。除了失败的握手请求响应，clientId字段必须在每个响应消息中中返回，clientId在消息到消息中是可选的。

**3.6. 建议advice**

The advice field provides a way for servers to inform clients of their preferred mode of client operation so that in conjunction with server-enforced limits, Bayeux implementations can prevent resource exhaustion and inelegant failure modes.

advice字段为服务端提供了一种通知客户端他们首选的客户端操作模式，因此，结合强加的服务端限制，Bayeux实现能避免资源耗尽和不雅的失败情形。

The advice field is a JSON map containing general and transport specific values that indicate modes of operation, timeouts and other potential transport specific parameters. Fields may occur either in the top level of an advice or within a transport specific section.

advice字段是一个JSON对象，包含了通用值和传输方式特定的值，有操作模式，超时，和其他潜在的特定传输方式参数。字段既可以在advice顶层也可以在传输方式特定段内。

Unless otherwise specified in sections 5 and 6, any Bayeux response message may contain an advice field. Advice received always superceeds any previous received advice.

An example advice field is

"advice": {   
"reconnect": "retry",  
"interval": 1000,  
"callback-polling" : {  
"reconnect": "handshake"  
}  
}  
除非是章节5和6指定的，其他任何Bayeux响应消息都可以包含一个advice字段。收到的advice总是替代之前接受的任何advice。  
advice字段示例如下：  
"advice": {   
     "reconnect": "retry",  
     "interval": 1000,  
     "callback-polling" : {  
         "reconnect": "handshake"  
     }  
}

**3.6.1. 重连advice reconnect advice**

The reconnect advice field is a string that indicates how the client should act in the case of a failure to connect. Defined reconnect values are:

重连advice字段是一个字符串，表明了客户端在连接失败的情况下应该如何操作。重连值定义有：

**retry**  
a client MAY attempt to reconnect with a /meta/connect after the interval (as defined by "interval" advice or client-default backoff), and with the same credentials.   
**重试**  
客户端可以在一定间隔（按照interval advice定义或客户端默认延时）后用同样的证书重连到/meta/connect。  
  
**handshake**  
the server has terminated any prior connection status and the client MUST reconnect with a /meta/handshake. A client MUST NOT automatically retry if handshake advice has been received.   
**握手**  
服务端已经终止所有之前的连接状态，客户端必须重连到/meta/handshake。如果客户端已经收到握手advice则禁止自动重试。  
  
**none**  
hard failure for the connect attempt. Do not attempt to reconnect at all. A client MUST respect reconnect advice of none and MUST NOT automatically retry or handshake. Any client that does not implement all defined values of reconnect MUST NOT automatically retry or handshake.  
**无**  
连接完全失败。不要再试图连接。客户端必须重视none重连advice，并禁止自动重试或握手。不实现全部reconnect定义值的客户端禁止自动重连或握手。

**3.6.2. 间隔advice interval advice**

An integer representing the minimum period in milliseconds for a client to delay subsequent requests to the /meta/connect channel. A negative period indicates that the message should not be retried.

A client MUST implement interval support, but a client MAY exceed the interval provided by the server. A client SHOULD implement a backoff strategy to increase the interval if requests to the server fail without new advice being received from the server.

一个提供给客户端的以秒为单位的整数所表示的最小周期，来推迟后续到/meta/connect通道的请求。

客户端必须实现间隔支持，但是一个客户端可以超过服务端提供的间隔。如果到服务端请求失败而没有收到新的advice，则客户端必须实现一种退避策略以增加间隔。

**3.6.3. 多客户端advice multiple-clients advice**

This is a boolean field, which if true indicates that the server has detected multiple Bayeux client instances running within the same HTTP client.

多客户端advice是一个布尔值字段，如果为真则表示服务端已经侦测到多个Bayeux客户端实例运行在同一个http客户端中。

**3.6.4. 主机advice hosts advice**

This is an array of strings field, which if present indicates a list of host names or IP addresses that MAY be used as alternate servers with which the client may connect. If a client receives advice to re-handshake and the current server is not included in a supplied hosts list, then the client SHOULD try the hosts in order until a successful connection is establish. Advice received during handshakes with hosts in the list supercedes any previously received advice.

主机advice是一个数组字符串字段，所表示的是一个主机名或IP地址列表，可作为候选服务器与客户端连接。如果一个客户端接收到重新握手的advice，而当前服务器不包含在提供的主机列表中，客户端就应该按顺序尝试这些主机，知道成功建立一个连接。和列表中的主机握手过程中收到的advice，要替代之前收到的任何advice。

**3.7. 连接类型 connectionType**

The connectionType message field specifies the type of transport the client requires for communication. The connectionType message field MUST be included in request messages to the "/meta/connect" channel. Connection types are defined in section 4.7.

连接类型消息字段指定了客户端用来通信所需的传输类型。连接类型消息字段必须包含在到/meta/connect通道的请求消息中。连接类型定影在章节4.7。

**3.8. id**

An id field MAY be included in any Bayeux message with an alpha numeric value:

id = alphanum \*( alphanum )

Generation of IDs is implementation specific and may be provided by the application. Messages published to /meta/\*\* and /service/\*\* SHOULD have id fields that are unique within the the connection.

Messages sent in response to messages delivered to /meta/\*\* channels MUST use the same message id as the request message.

Messages sent in response to messages delivered to /service/\*\* channels SHOULD use the same message id as the request message or an id derived from the request message id.

一个字符数字值的id字段可以包含在任何Bayeux消息中：

id = alphanum \*( alphanum )

ID的生成是实现相关的，也可有应用提供。发布到/meta/\*和/service/\*的消息，应该好汉id字段，并在连接内唯一。

发送到 /meta/\*\* 通道的消息的响应消息，必须使用和请求消息同样的消息id。

发送到 /service/\*\*通道的消息的响应消息，应该使用和请求消息同样的消息id，或者派生自请求消息id。

**3.9. 时间戳 timestamp**

The timestamp message field SHOULD be specified in the following ISO 8601 profile: All times SHOULD be sent in GMT time.

YYYY-MM-DDThh:mm:ss.ss  
A timestamp message field is OPTIONAL in all Bayeux messages.  
时间戳消息字段应该指定为下面的ISO8601格式：所有时间应该用GMT时间发送。  
YYYY-MM-DDThh:mm:ss.ss  
时间戳消息字段在任何Bayeux消息中都是可选的。

**3.10. 数据 data**

The data message field is an arbitrary JSON object that contains event information. The data field MUST be included in publish requests, and a Bayeux server MUST include the data field in an event delivery message.

数据消息字段是一个包含时间信息的任意的JSON对象。数据字段必须包含在publish请求中，并且Bayeux服务端也必须在时间发送消息中包含数据字段。

**3.11. 连接Id connectionId**

The connectionId field was used during development of the Bayeux protocol and its use is now deprecated.

连接Id字段在Bayeux开发阶段使用，现在已经不推荐使用。

**3.12. 成功的 successful**

The successful boolean message field is used to indicate success or failure and MUST be included in responses to the "/meta/handshake", "/meta/connect", "/meta/subscribe","/meta/unsubscribe", "/meta/disconnect", and publish channels.

成功布尔消息字段用来表明成功或失败，必须包含在到"/meta/handshake", "/meta/connect", "/meta/subscribe","/meta/unsubscribe", "/meta/disconnect", 和publish通道消息的响应中。

**3.13. 订阅 subscription**

The subscription message field specific the channels the client wishes to subscribe to or unsubscribe from. The subscription message field MUST be included in requests and responses to/from the "/meta/subscribe" or "/meta/unsubscribe" channels.

订阅消息字段明确了客户端想订阅的和接触订阅的通道。订阅消息字段必须包含在"/meta/subscribe"通道的请求中，或者"/meta/unsubscribe"通道的响应中。

**3.14. 错误 error**

The error message field is OPTIONAL on any Bayeux response. The error message field MAY indicate the type of error that occurred when a request returns with a false successful message. The error message field should be sent as a string in the following format:

error = error\_code ":" error\_args ":" error\_message   
| error\_code ":" ":" error\_message  
error\_code = digit digit digit  
error\_args = string \*( "," string )  
error\_message = string

Example error strings are:

401::No client ID  
402:xj3sjdsjdsjad:Unknown Client ID  
403:xj3sjdsjdsjad,/foo/bar:Subscription denied  
404:/foo/bar:Unknown Channel  
Need to provide list of codes  
错误消息字段在任何Bayeux响应中都是可选的。当一个请求返回不成功消息时，错误消息字段可以表明发生的错误类型。错误消息字段应该按照以下格式的字符串被发送：  
error = error\_code ":" error\_args ":" error\_message   
| error\_code ":" ":" error\_message  
error\_code = digit digit digit  
error\_args = string \*( "," string )  
error\_message = string

Example error strings are:

401::No client ID  
402:xj3sjdsjdsjad:Unknown Client ID  
403:xj3sjdsjdsjad,/foo/bar:Subscription denied  
404:/foo/bar:Unknown Channel  
Need to provide list of codes

**3.15. 扩展 ext**

An ext field MAY be included in any Bayeux message. Its value SHOULD be a JSON map with top level names distinguished by implementation names (eg. "org.dojo.Bayeux.field").

The contents of ext may be arbitrary values that allow extensions to be negotiated and implemented between server and client implementations.

扩展字段可以包含在任何Bayeux消息中，它的值应该是以实现名称显著显示的顶级名称（如："org.dojo.Bayeux.field"）的JSON map。扩展内容可以是任意值，这些值允许扩展被协商和实现在服务端和客户端实现之间。

**3.16. json-comment-filtered**

The json-comment-filtered ext field of the handshake message is deprecated.  
握手消息的json-comment-filtered扩展字段已经不推荐使用。

**4. Meta消息定义 Meta Message Definitions**

**4.1. 握手 handshake**

**4.1.1. 握手请求 handshake Request**

A Bayeux client initiates a connection negotiation by sending a message to the "/meta/handshake" channel. For same domain connections, the Handshake requests MUST be sent to the server as the 'message' parameter of an application/x-www-form-urlencoded encoded POST request. For cross domain connections, the Handshake request MUST be sent to the server as a url encoded GET request with the jsonp parameter set for callback-polling.

一个Bayeux客户端往/meta/handshake发送一个消息来开始一个连接协商。对同域连接，握手请求必须做为application/x-www-form-urlencoded编码的POST请求的message参数发送到服务端。对跨域连接，握手请求必须作为url编码的GET请求的为回调轮询设置的jsonp参数发往服务端。

A handshake request MUST contain the message fields:

**channel**  
value "/meta/handshake"   
**version**  
The version of the protocol supported by the client.   
**supportedConnectionTypes**  
An array of the connection types supported by the client for the purposes of the connection being negotiated. This list MAY be a subset of the connection types actually supported if the client wishes to negotiate a specific connection type.  
一个握手请求必须包含的消息字段：  
**channel**  
value "/meta/handshake"   
值为 /meta/handshake  
**version**  
The version of the protocol supported by the client.   
客户端支持的协议版本  
**supportedConnectionTypes**  
An array of the connection types supported by the client for the purposes of the connection being negotiated. This list MAY be a subset of the connection types actually supported if the client wishes to negotiate a specific connection type.   
客户端支持的一个连接类型数组，用做连接协商。如果客户端想协商为一种特定的连接类型，则支持的连接类型列表可以是实际支持连接类型的子列表。

A handshake request MAY contain the message fields:

一个握手请求可以包含的消息字段：

**minimumVersion**  
**ext**  
**id**

A client SHOULD NOT send any other message in the request with a handshake message. A server MUST ignore any other message sent in the same request as a handshake message. An example handshake request is:

[  
{  
"channel": "/meta/handshake",  
"version": "1.0",  
"minimumVersion": "1.0beta",  
"supportedConnectionTypes": ["long-polling", "callback-polling", "iframe"]  
}  
]  
客户端不应该在握手消息的请求中发送其他消息。服务端必须忽略和握手消息中同一个请求中的其他消息。  
握手请求示例如下：  
[  
{  
"channel": "/meta/handshake",  
"version": "1.0",  
"minimumVersion": "1.0beta",  
"supportedConnectionTypes": ["long-polling", "callback-polling", "iframe"]  
}  
]

**4.1.2. 握手响应 handshake Response**

A Bayeux server MUST respond to a handshake request with a handshake response message in the body content of the response. For cross domain connections that have the 'jsonp' parameter set, the message body may be encapsulated in a jsonp callback method.

Bayeux服务端必须在响应体内容中包含握手响应消息对握手请求作出响应。对设置jsonp参数的跨域连接，消息体必须封装在json回调方法中。

**Successful handshake response**

**成功的握手响应**

A successful handshake responses MUST contain the message fields:

一个成功的握手影响必须包含以下消息字段：

**channel**  
value "/meta/handshake"  
值为 /meta/handshake  
   
**version**  
**supportedConnectionTypes**  
The connection types supported by the server for the purposes of the connection being negotiated. This list MAY be a subset of the connection types actually supported if the server wishes to negotiate a specific connection type. This list MUST contain at list one element in common with the supportedConnectionType provided in the handshake request. If there are no connectionTypes in common, the handshake response MUST be unsuccessful.  
服务端支持的连接类型，用作连接协商。如果服务端想协商为一种特定的连接类型，则连接支持列表可以是实际支持的子列表。服务端连接支持列表必须至少包含一种，和握手请求中支持连接类型中所提供的相同。如果没有共有的连接类型，握手响应就必须是不成功的。  
   
**clientId**  
A newly generated unique clientId.  
一个新产生的唯一的clientId。  
   
**successful**  
value true  
值为真

A successful handshake response MAY contain the message fields:

一个成功的握手响应可以包含的消息字段：

**minimumVersionadvice  
ext  
id**  
same value as request message id   
和请求消息id相同  
  
**authSuccessful**   
Value true, this field may be included to support prototype client implementations that required the authSuccessful field  
值为真，这个字段可以包含，以支持需要authSuccessful字段的原型客户端实现。  
   
An example successful handshake response is: [  
{  
"channel": "/meta/handshake",  
"version": "1.0",  
"minimumVersion": "1.0beta",  
"supportedConnectionTypes": ["long-polling","callback-polling"],  
"clientId": "Un1q31d3nt1f13r",  
"successful": true,  
"authSuccessful": true,  
"advice": { "reconnect": "retry" }  
}  
]

**Unsuccessful handshake response**

不成功握手响应

An unsuccessful handshake response MUST contain the message fields:

不成功的握手响应必须包含的消息字段：

**channel**  
value "/meta/handshake"   
值为/meta/handshake  
**successful**  
value false   
值为false  
**error**  
a string with the description of the reason for the failure.  
失败原因的描述字符串

An unsuccessful handshake response MAY contain the message fields:

一个不成功的握手响应可以包含的消息字段：

**supportedConnectionTypes**  
The connection types supported by the server for the purposes of the connection being negotiated. This list MAY be a subset of the connection types actually supported if the server wishes to negotiate a specific connection type.   
服务端支持的连接类型，用作连接协商。如果服务端想协商为一种特定的连接类型，则连接支持列表可以是实际支持的子列表。  
**advice**  
**version**  
**minimumVersion**  
**ext**  
**id**  
same value as request message id  
和请求消息id相同

An example unsuccessful handshake response is:

一个不成功的握手响应例子为：

[  
{  
"channel": "/meta/handshake",  
"version": "1.0",  
"minimumVersion": "1.0beta",  
"supportedConnectionTypes": ["long-polling","callback-polling"],  
"successful": false,  
"error": "Authentication failed",  
"advice": { "reconnect": "none" }  
}  
]

For complex connection negotiations, multiple handshake messages may be exchanged between the Bayeux client and server. The handshake response will set the "successful" field to false until the handshake processs is complete. The advice and ext fields may be used to communicate additional information needed to complete the handshake process. An unsuccessful handshake response with reconnect advice of "handshake" is used to continue the connection negotiation. An unsuccessful handshake response with reconnect advice of "none" is used to terminate connection negotiations.

对于复杂的连接协商，Bayeux客户端和服务端之间可能有多个握手消息交互。握手响应会设置successful字段为false，直到握手过程完成。advice和ext字段可以用来传送完成握手过程所需的额外信息。一个带有重连advice“handshake”的不成功的握手响应常用来继续连接协商。一个带有重连advice“none”的不成功的握手响应，常用来终止连接协商。

**4.2. 连接 connect**

**4.2.1. 连接请求 connect Request**

After a Bayeux client has discovered the server's capabilities with a handshake exchange, a connection is established by sending a message to the "/meta/connect" channel. This message may be transported over any of the transports indicated as supported by the server in the handshake response.

经过握手交互一个Bayeux客户端发现了服务端的能力后，依靠发送给“/meta/connect”通道一个消息来建立一个连接。这个消息可以使用握手响应中服务端表明所支持的任何传输方式来传输。

A connect request MUST contain the message fields:

一个连接请求必须包含的消息字段：

**channel**  
value "/meta/connect"   
值为“/meta/connect”  
**clientId**  
The client ID returned in the handshake response   
握手响应中返回的客户端ID  
**connectionType**  
The connection type used by the client for the purposes of this connection.  
客户端连接所使用的连接类型。

A connect request MAY contain the message fields:

一个连接请求可能包含的消息字段：

**ext**  
**id**

A client MAY send other messages in the same HTTP request with a connection message. A server MUST handle any other message sent in the same request as a connect message after the handling of the connect message is complete.

一个客户端可以在和一个连接消息同样的HTTP请求中发送其他消息。服务端必须在连接消息处理完成后，处理和连接消息同样请求中发送的其他任何消息。

An example connect request is:

一个连接请求的例子为：

[  
{  
"channel": "/meta/connect",  
"clientId": "Un1q31d3nt1f13r",  
"connectionType": "long-polling"  
}  
]

A transport MUST maintain one and only one outstanding connect message. When a HTTP response that contains a /meta/connect response terminates, the client MUST wait at least the interval specified in the last received advice before following the advice to reestablish the connection

一个传输通道必须维持一个且仅有一个未决连接信息。当一个HTTP响应包含了一个/meta/connect响应终止时，客户端必须至少等待上一个收到的advice指定的间隔后，才能继续建议连接。

**4.2.2. 连接响应 connect Response**

A Bayeux server MUST respond to a connect request with a connect response message over the same transport as used for the request.

Bayeux服务端必须基于和请求相同的传输方式，用请求响应消息对连接请求做出响应。

A Bayeux server MAY wait to respond until there are event messages available in the subscribed channels for the client that need to be delivered to the client.

Bayeux服务器可以一直等待，直到客户端订阅的通道上有事件消息需要发送到客户端时，再做出响应。

A connect responses MUST contain the message fields:

一个连接响应必须包含的消息字段：

**channel**  
value "/meta/connect"  
值为/meta/connect  
**successful**  
boolean indicating the success or failure of the connection  
指示连接失败或成功的布尔值  
**clientId**  
The negotiated client ID  
协商的客户端ID

A connect response MAY contain the message fields:

一个连接响应可能包含的消息字段：

**error  
advice  
ext  
id**  
same value as request message id  
和请求消息id相同  
**timestamp**

An example connect response is:

连接响应的一个例子：

[  
{  
"channel": "/meta/connect",  
"successful": true,  
"error": "",  
"clientId": "Un1q31d3nt1f13r",  
"timestamp": "12:00:00 1970",  
"advice": { "reconnect": "retry" }  
}  
]

The client MUST maintain only a single outstanding connect message. If the server does not have a current outstanding connect and a connect is not received within a configured timeout, then the server SHOULD act as if a disconnect message has been received.

客户端必须维护唯一的未决连接消息。如果服务端没有一个当前未决的连接，并且在配置的超时时间内，没有收到一个连接，服务端应该认为收到了一个断开消息。

**4.4. 断开 disconnect**

**4.4.1. 断开请求 disconnect Request**

When a connected client wishes to cease operation it should send a request to the "/meta/disconnect" channel for the server to remove any client-related state. The server SHOULD release any waiting meta message handlers. Bayeux client applications should send a disconnect request when the user shuts down a browser window or leaves the current page. A Bayeux server SHOULD not rely solely on the client sending a disconnect message to remove client-related state information because a disconnect message might not be sent from the client or the disconnect request might not reach the server.

一个已连接客户端想终止操作的时候，它应该给/meta/disconnect通道发送一个请求，让服务端移除任何客户端相关的状态。服务端应该释放任何等待中的元消息处理。当用户关闭浏览器窗口或者离开当前页面时，Bayeux客户端应用应该发送一个断开请求。Bayeux服务端不应该只依赖于客户端发送断开消息才移除客户端相关的状态信息，因为断开消息可能不从客户端发送或者断开请求不会到达服务端。

A disconnect request MUST contain the message fields:   
一个断开请求必须包含的消息字段：  
**channel**  
value "/meta/disconnect"   
值为/meta/disconnect  
**clientId**  
The client ID returned in the handshake response  
握手响应中返回的客户端ID

A disconnect request MAY contain the message fields:

一个断开请求可能包含的消息字段：

**ext  
id**  
An example disconnect request is:   
断开请求的一个例子：  
[  
{  
"channel": "/meta/disconnect",  
"clientId": "Un1q31d3nt1f13r"  
}  
]

**4.4.2. 断开响应 disconnect Response**

A Bayeux server MUST respond to a disconnect request with a disconnect response.

Bayeux服务端必须用一个断开响应对一个断开请求作出响应。

A disconnect response MUST contain the message fields:

一个断开响应必须包含的消息字段：

**channel**  
value "/meta/disconnect"   
值为/meta/disconnect  
**clientId**  
The client ID returned in the handshake response   
握手响应中返回的客户端ID  
**successful**  
boolean value indicated the success or failure of the disconnect request  
表明断开请求成功或失败的布尔值

A disconnect response MAY contain the message fields:

一个断开响应可能包含的消息字段：

**error  
ext  
id**  
same value as request message id   
和请求消息id相同  
  
An example disconnect response is:   
一个断开响应的例子：  
[  
{  
"channel": "/meta/disconnect",  
"clientId": "Un1q31d3nt1f13r",  
"successful": true   
}  
]

**4.5. 订阅 subscribe**

**4.5.1. 订阅请求 subscribe Request**

A connected Bayeux client may send subscribe messages to register interest in a channel and to request that messages published to the subscribe channel are delivered to the client.

一个已连接Bayeux客户端可以发送订阅消息，登记关心的一个通道，并请求发布到这个订阅通道的消息被发送到这个客户端。

A subscribe request MUST contain the message fields:

一个订阅请求必须包含的消息字段：

**channel**  
value "/meta/subscribe"   
值为“/meta/subscribe”  
**clientId**  
The client ID returned in the handshake response   
握手响应中返回的客户端ID  
**subscription**  
a channel name or a channel pattern or an array of channel names and channel patterns.  
一个通道名称或者一个通道模式或者一个通道名称和通道模式的数组

A subscribe request MAY contain the message fields:

一个订阅请求可能包含的消息字段：

**ext  
id**

An example subscribe request is:

一个订阅请求的例子：

[  
{  
"channel": "/meta/subscribe",  
"clientId": "Un1q31d3nt1f13r",  
"subscription": "/foo/\*\*"  
}  
]

**4.5.2. 订阅响应 subscribe Response**

A Bayeux server MUST respond to a subscribe request with a subscribe response message.

Bayeux服务端必须用订阅响应消息对订阅请求作出响应。

A Bayeux server MAY send event messages for the client in the same HTTP response as the subscribe response, including events for the channels just subscribed to.

Bayeux服务端可以在和订阅响应同样的HTTP响应中给客户端发送事件消息，包括所订阅通道的事件。

A subscribe responses MUST contain the message fields:

一个订阅响应必须包含的消息字段：

**channel**  
value "/meta/subscribe"  
值为"/meta/subscribe"  
**successful**  
boolean indicating the success or failure of the subscribe  
表明订阅成功或失败的布尔值  
**clientId**  
The negotiated client ID  
协商的客户端ID  
**subscription**  
a channel name or a channel pattern or an array of channel names and channel patterns.   
一个通道名称或者一个通道模式或者一个通道名称和通道模式的数组  
  
A subscribe response MAY contain the message fields:   
一个订阅响应可能包含的消息字段： **error  
advice  
ext  
id**  
same value as request message id  
和请求消息id相同  
**timestamp**  
An example successful subscribe response is:   
一个成功的订阅响应例子：  
[  
{  
"channel": "/meta/subscribe",  
"clientId": "Un1q31d3nt1f13r",  
"subscription": "/foo/\*\*",  
"successful": true,  
"error": ""  
}  
]  
An example failed subscribe response is:   
一个失败的订阅响应例子：  
[  
{  
"channel": "/meta/subscribe",  
"clientId": "Un1q31d3nt1f13r",  
"subscription": "/bar/baz",  
"successful": false,  
"error": "403:/bar/baz:Permission Denied"  
}  
]

**4.6. 取消订阅 unsubscribe**

**4.6.1. 取消订阅请求 unsubscribe Request**

A connected Bayeux client may send unsubscribe messages to cancel interest in a channel and to stop published message delivery from the server to the unsubscribe channel.

一个已连接Bayeux客户端可以发送取消订阅消息，取消关心的一个通道，并停止从服务端投递到取消订阅通道的发布消息。

A unsubscribe request MUST contain the message fields:

一个取消订阅请求必须包含的消息字段：

**channel**  
value "/meta/unsubscribe"   
值为"/meta/unsubscribe"   
**clientId**  
The client ID returned in the handshake response   
握手响应中返回的客户端ID  
**subscription**  
a channel name or a channel pattern or an array of channel names and channel patterns.  
一个通道名称或者一个通道模式或者一个通道名称和通道模式的数组

A unsubscribe request MAY contain the message fields:

一个取消订阅请求可能包含的消息字段：

**ext  
id**  
An example unsubscribe request is:   
一个取消订阅请求的例子：  
[  
{  
"channel": "/meta/unsubscribe",  
"clientId": "Un1q31d3nt1f13r",  
"subscription": "/foo/\*\*"  
}  
]

**4.6.2. 取消订阅响应 unsubscribe Response**

A Bayeux server MUST respond to a unsubscribe request with a unsubscribe response message.

Bayeux服务端必须用取消订阅响应消息对取消订阅请求做出响应。

A Bayeux server MAY send event messages for the client in the same HTTP response as the unsubscribe response, including events for the channels just unsubscribed to as long as the event was processed before the unsubscribe request.

Bayeux服务端可以在和取消订阅响应同样的HTTP响应中给客户端发送事件消息，包括取消订阅通道的事件，只要事件在取消请求前处理。

A unsubscribe responses MUST contain the message fields:   
一个取消订阅响应必须包含的消息字段：  
**channel**  
value "/meta/unsubscribe"  
值为"/meta/unsubscribe"  
**successful**  
boolean indicating the success or failure of the unsubscribe operation  
表明取消订阅操作成功或失败的布尔值  
**clientId**  
The negotiated client ID  
**subscription**  
a channel name or a channel pattern or an array of channel names and channel patterns.   
一个通道名称或者一个通道模式或者一个通道名称和通道模式的数组  
  
A unsubscribe response MAY contain the message fields:   
一个取消订阅响应可能包含的消息字段：  
**error  
advice  
ext  
id**  
same value as request message id  
和请求消息id相同  
**timestamp**  
An example unsubscribe response is:   
一个取消订阅响应的例子：  
[  
{  
"channel": "/meta/unsubscribe",  
"clientId": "Un1q31d3nt1f13r",  
"subscription": "/foo/\*\*",  
"successful": true,  
"error": ""  
}  
]

**5. 事件消息定义 Event Message Definitions**

Application events are published in event messages sent from a Bayeux client to a Bayeux server and are delivered in event messages sent from a Bayeux server to a Bayeux client.  
应用事件通过Bayeux客户端发送到Bayeux服务端的事件消息发布，并通过Bayeux服务端发送到Bayeux客户端的事件消息传送。

**5.1. 发布事件消息 Publish event messages**

**5.1.1. 发布请求 publish Request**

A Bayeux client can publish events on a channel by sending event messages. An event message MAY be sent in new HTTP request or it MAY be sent in the same HTTP request as any message other than a handshake meta message.

Bayeux客户端能在通道上发送事件消息发布事件。一个事件消息可以在新的HTTP请求中发送，或者可以和除了握手元消息之外的任何消息相同的HTTP请求中发送。

A publish message may be sent from an unconnected client (that has not performed handshaking and thus does not have a client ID). It is OPTIONAL for a server to accept unconnected publish requests and they should apply server specific authentication and authorization before doing so.

一个发布消息可以从一个未连接的客户端发送（还没有握手因此没有客户端ID）。服务端接受未连接的发布请求是可选的，因此他们应该在这么做之前应用服务端特定的认证和授权。

A publish event message MUST contain the message fields:   
一个发布事件消息必须包含的消息字段：  
**channel  
data**The message as an arbitrary JSON object   
消息可以是任何的JSON对象  
  
A publish event message MAY contain the message fields:   
一个发布事件消息可能包含的消息字段：  
**clientId**  
The negotiated client ID  
协商的客户端ID  
**id**  
A unique ID for the message generated by the client  
客户端生成的唯一消息ID  
**ext**  
  
An example event message is:   
一个事件消息的例子：  
[  
{  
"channel": "/some/channel",  
"clientId": "Un1q31d3nt1f13r",  
"data": "some application string or JSON object",  
"id": "some unique message id"  
}  
]

**5.1.2. 发布响应 publish Response**

A Bayeux server MAY respond to a publish event message with a publish event acknowlegement.

服务端可以用发布事件确认对发布事件消息做出响应。

A publish event message MUST contain the message fields:   
一个发布事件消息必须包含的消息字段：  
**channel**  
**successful**  
boolean indicating the success or otherwise of the publish   
表明发布成功或其它的布尔值  
  
A publish event response MAY contain the message fields:   
一个发布事件响应可能包含的消息字段：  
**id  
error  
ext**  
An example event reponse message is:   
一个事件响应消息的例子：  
[  
{  
"channel": "/some/channel",  
"successful": true,  
"id": "some unique message id"  
}  
]

**5.2. 传送事件消息 Deliver Event messages**

Event messages are delivered to clients if the client is subscribed to the channel of the event message. Event messages may be sent to the client in the same HTTP response as any other message other than a meta handshake response. If a Bayeux server has multiple HTTP requests from the same client, the server SHOULD deliver all available messages in the HTTP response that will be sent immediately in preference to waking a waiting connect meta message handler. Event message delivery is not acknowledged by the client.

如果客户端订阅到事件消息的通道，事件消息就可以被传送到客户端。事件消息可以在除了元握手响应之外的任何消息相同的HTTP响应中被发往客户端。如果Bayeux服务端有来自同一客户端的多个HTTP请求，服务端应该在HTTP响应中传送所有可用消息，TODO

事件消息传送不必被客户端确认。

A deliver event message MUST contain the message fields:

一个传送事件消息必须包含的消息字段：

**channel  
data**  
The message as an arbitrary JSON object   
消息可以是任何的JSON对象  
  
A deliver event response MAY contain the message fields:   
一个传送事件响应可能包含的消息字段：  
**id**  
Unique message ID from the publisher  
来自发布者的唯一消息ID  
**clientId**  
The client ID of the publisher  
发布者的客户端ID  
**ext  
advice**An example event deliver message is:   
一个事件传送消息的例子：  
[  
{  
"channel": "/some/channel",  
"data": "some application string or JSON object",  
"id": "some unique message id"  
}  
]

**6. 传输 Transports**

**6.1. 长轮询 long-polling**

"Long-polling" is a polling transport that attempts to minimize both latency in server-client message delivery, and the processing/network resources required for the connection. In "traditional" polling, servers send and close responses to requests immediately, even when there are no events to deliver, and worst-case latency is the polling delay between each client request. Long-polling server implementations attempt to hold open each request until there are events to deliver; the goal is to always have a pending request available to use for delivering events as they occur, thereby minimizing the latency in message delivery. Increased server load and resource starvation are addressed by using the reconnect and interval advice fields to throttle clients, which in the worst-case degenerate to traditional polling behaviour.  
长轮询是一种轮询方式，这种方式试图最小化服务端-客户端消息传送延迟和连接的处理/网络资源消耗。传统的轮询中，服务端发送并立即关闭到请求的响应，尤其是没有事件传送时，并且最糟糕情况下的延时就是每个客户端请求之间轮询的耽搁。长轮询服务端实现试图保持打开每一个请求，直到有事件要传送；目的就是总有一个可用的未决请求在有事件发生时传送事件，因此最小化了消息传送的延时。靠使用重连和间隔建议字段应对增加的服务端负载和资源竞争，进而节制客户端，最糟糕的情况就是退化到传统的轮询方式。

**6.1.1 长轮询请求消息 long-polling request messages**

Messages are sent to the server as the body of a POST, encoded either as "application/x-www-form-urlencoded" or as "text/json". If sent as form encoded, the Bayeux messages are sent as the "message" parameter in one of the following forms as:

消息做为POST请求体发往服务端，可以被编码为 "application/x-www-form-urlencoded" 或 "text/json"。如果做为表单编码发送，Bayeux消息做为下列形式之一的"message"参数发送：

* Single valued and contain a single Bayeux message
* 单个值并包含单个Bayeux消息
* Single valued and contain an array of Bayeux message
* 单个值并包含一个Bayeux消息数组
* Multi valued and contain a several individual Bayeux message
* 多值并包含多个独立的Bayeux消息
* Multi valued and contain a several arrays of Bayeux message
* 多值并包含多个Bayeux消息数组
* Multi valued and contain a mix of individual Bayeux messages and arrays of Bayeux message
* 多值并混合包含了独立的Bayeux消息和Bayeux消息数组

**6.1.2 长轮询响应消息 long-polling response messages**

Messages are sent to the client as unencapsulated body content of a POST response with content type "text/json" or "text/json-comment-filtered".

消息做为未封装的POST响应体发往客户端，POST响应的内容类型为"text/json" 或 "text/json-comment-filtered"。

**6.2. 回调轮询 callback-polling**

**6.2.1 回调轮询请求消息 callback-polling request messages**

Messages are sent to the server either using POST requests as per long-polling transport or as the 'message' URL parameter of a GET request.

发送到服务端的消息，既可以使用每个长轮询传输的POST请求，也可以做为GET请求的message参数。

**6.2.2 回调轮询响应消息 callback-polling response messages**

Messages are sent to the client as JavaScript function call returned for script source GET requests. The function called will be determined by the 'jsonp' field of any associated request messages, or 'jsonpcallback' if not specified. The called function will be passed a JSON array of Bayeux messages.

发送到客户端的消息，当作JavaScript函数调用为脚本源GET请求而返回。被调用的函数由相关请求消息的jsonp字段决定，如果不指定（jsonp参数）函数就是jsonpcallback。调用的函数被传入一个Bayeux消息的JSON数组。

**7. 安全 Security**

**7.1. 认证 Authentication**

Bayeux may be used with:  
Bayeux可以使用

* No authentication
* 无认证
* Container supplied authentication (eg BASIC auth or cookie managed session based authentication)
* 容器提供的认证（如BASIC认证或基于session的cookie管理的认证）
* Bayeux extension authentication that exchanges authentication credentials and tokens within Bayeux messages ext fields
* 在Bayeux消息扩展字段中交换认证证书和令牌的Bayeux扩展认证

For Bayeux authentication, no algorithm is specified for generating or validating security credentials or token. This version of the protocol only defines that the ext field may be used to exchange authentication challenges, credentials, and tokens and that the advice field may be used to control multiple iterations of the exchange.

Bayeux认证并没有指定算法，来产生或验证安全证书或令牌。协议的这个版本只定义了扩展字段可被用来交换认证口令，证书，和令牌，而advice字段可以用来控制交换的多次迭代。

The connection negotiation mechanism may be used to negotiate authentication or request re-authentication.

连接协商机制可以被用来协商认证或请求重新认证。

**7.2. Ajax劫持 Ajax Hijacking**

The Ajax hijacking vulnerability is when an attacking web site uses a script tag to execute JSON content obtained from an Ajax server. The Bayeux protocol is not vulnerable to this style of attack as cookies are not used for authentication and a valid client ID is needed before private client data is returned. The use of POST by some transports further protects against this style of attack.

Ajax劫持漏洞是当攻击网站使用script标记来是执行从Ajax服务端获取的JSON内容。Bayeue协议不易受这种攻击，因为不使用cookie来认证，并且在私有客户端数据返回前需要一个有效的客户端ID。某些使用POST的传输方式更进一步避免这种类型的攻击。

**8. 多框架运作 Multi frame operation**

Current HTTP client implementations are RECOMMENDED to allow only two connections between a client and a server. This presents a problem when multiple instances of the Bayeux client are operating in multiple tabs or windows of the same browser instance. The two connection limit can be consumed by outstanding connect meta messages from each tab or window and thus prevent other messages from being delivered in a timely fashion.

当前的HTTP客户端实现只允许在一个客户端和一个服务端之间两个连接。这在多Bayeux客户端实例在同一个浏览器实例的多个标签或窗口中操作时，表现出一个问题。两个连接的限制被来自每个标签或窗口的未决的连接元消息所耗尽，从而防止其他消息被及时传递。

**8.1 服务端多框架侦测 Server Multi frame detection**

It is RECOMMENDED that Bayeux server implementations use the cookie "Bayeux\_HTTP\_ID" to identify a HTTP client and to thus detect multiple Bayeux clients running within the same HTTP client. Once detected, the server SHOULD not wait for messages in connect and SHOULD use the advice interval mechanism to establish traditional polling.

推荐Bayeux服务端实现用cookie"Bayeux\_HTTP\_ID"来标识一个HTTP客户端，以此来侦测同一个HTTP客户端内的多个Bayeux客户端。一旦侦测到，服务端不应该等待连接的消息，应该使用建议间隔机制来建立传统的轮询。

**8.2 客户端多框架处理 Client Multi frame handling**

It is RECOMMENDED that Bayeux client implementations use client side persistence or cookies to detect multiple intances of Bayeux clients running within the same HTTP client. Once detected, the user MAY be offered the option to disconnect all but one of the clients. It MAY be possible for client implementations to use client side persistence to share a Bayeux client instance.

推荐Bayeux客户端实现使用客户侧的存储或cookie来侦测同一个HTTP客户端内的多Bayeux客户端实例。一旦发现，用户可以选择断开所有客户端而只保留一个。客户端实现也可能用客户端侧的持久化来共享一个Bayeux客户端实例。

**9. service通道的请求/响应操作**

**Request / Response operation with service channels**

The publish/subscribe paradigm that is directly supported by the Bayeux protocol is difficult to use to efficiently implement the request/response paradigm between a client and a server. The /service/\*\* channel space has been designated as a special channel space to allow efficient transport of application request and responses over Bayeux channels. Messages published to service channels are not distributed to other Bayeux clients so these channels can be used for private requests between a Bayeux client and a server side handlers.

A trivial example would be an echo service, that sent any message received from a client back to that client unaltered. Bayeux clients would subscribe the the /service/echo channel, but the Bayeux server would not need to record this subscription. When a client publishes a message to the /service/echo channel, it will be delivered only to server-side subscribers (in an implementation depedent fashion). The server side handler for the echo service would handle each message received by publishing a response directly to the client regardless of any subscription. As the client has subscribed to /service/echo, the response message will be routed correctly within the client to the appropriate application handler.

发布订阅模式被Bayeux协议直接支持，但在客户端和服务端之间有效的实现请求响应模式就比较困难。/service/\*\*通道空间被指定为一个特殊的通道空间，以允许基于Bayeux通道的应用请求响应的有效传输。发布到service通道的消息不被分发到其他Bayeux客户端，因此，这些通道可用作Bayeux客户端和服务端处理间的私有请求。  
举一个回显服务例子，发送的任何消息原封不动的返回到那个客户端。Bayeux客户端可以订阅/service/echo通道，但是Bayeux服务端不必记录这个订阅。当客户端发布一个到/service/echo通道的消息时，仅仅会被投送到服务端的订阅者（使用实现相关的方式）。echo服务的服务端处理会直接发布接收消息的响应到客户端，而不关心任何订阅。因为客户端已经订阅到/service/echo，响应消息会被无误的路由，在客户端中应用适当的处理。

[新用户注册](http://www.blogjava.net/RequireRegister.aspx)  [刷新评论列表](javascript:__doPostBack('AjaxHolder$PostComment$refreshList',''))

|  |  |  |
| --- | --- | --- |
|  | | |
|  | | |
|  |  |  |
|  | [找优秀程序员，就在博客园](http://job.cnblogs.com/) | |
| 标题 |  |  |
| 姓名 |  |  |
| 主页 |  |  |
| |  |  |  | | --- | --- | --- | |  | | | | 验证码 | \* |  | | | |
| 内容(请不要发表任何与政治相关的内容) | | |
|  | | |
| Remember Me? | | |
|  | [登录](http://www.blogjava.net/login.aspx?ReturnURL=http://www.blogjava.net/menlong999/archive/2009/08/16/291365.html&SourceURL=/menlong999/archive/2009/08/16/291365.html) | |
| [使用Ctrl+Enter键可以直接提交] | | |
|  | | |
|  | | |
|  | | |
|  | | |
| 网站导航:  [博客园](http://www.cnblogs.com/)   [IT新闻](http://news.cnblogs.com/)   [知识库](http://kb.cnblogs.com/)   [C++博客](http://www.cppblog.com/)   [程序员招聘](http://job.cnblogs.com/)   [管理](http://www.blogjava.net/menlong999/archive/2009/08/16/291365.html?opt=admin) | | |
|  | | |
|  | | |
|  | | |
|  | | |

**公告**

雁过留声，人过留名，既然来看了，就请留点只言片语。。

**常用链接**

* [我的随笔](http://www.blogjava.net/menlong999/MyPosts.html)
* [我的评论](http://www.blogjava.net/menlong999/MyComments.html)
* [我的参与](http://www.blogjava.net/menlong999/OtherPosts.html)
* [最新评论](http://www.blogjava.net/menlong999/RecentComments.html)

**评论排行榜**

* [1. Java、Linux、C++电子书分享(1)](http://www.blogjava.net/menlong999/archive/2008/09/28/231737.html)
* [2. Servlet与模板方法模式(0)](http://www.blogjava.net/menlong999/archive/2008/09/17/229332.html)
* [3. 不能在Google Docs上直接发布到blogjava了？？(0)](http://www.blogjava.net/menlong999/archive/2008/09/17/229285.html)
* [4. 移动硬盘与适配器模式(0)](http://www.blogjava.net/menlong999/archive/2008/09/15/229011.html)
* [5. Head First Pattern之单例模式(0)](http://www.blogjava.net/menlong999/archive/2008/09/14/228893.html)

**阅读排行榜**

* [1. Bayeux协议(2088)](http://www.blogjava.net/menlong999/archive/2009/08/16/291365.html)
* [2. Servlet与模板方法模式(694)](http://www.blogjava.net/menlong999/archive/2008/09/17/229332.html)
* [3. Java、Linux、C++电子书分享(394)](http://www.blogjava.net/menlong999/archive/2008/09/28/231737.html)
* [4. Head First Pattern之代理模式(368)](http://www.blogjava.net/menlong999/archive/2008/09/13/228810.html)
* [5. Head First Pattern之单例模式(279)](http://www.blogjava.net/menlong999/archive/2008/09/14/228893.html)

posts - 11, comments - 2, trackbacks - 0, articles - 0

Copyright © Lv Yuanfang