Project JXTA:
Technical Specification
Version 1.0

Introduction

What is JXTA?

JXTA is an open network computing *platform* designed for peer-to-peer (P2P) computing.

The JXTA platform standardizes the manner in which peers:

- Discover each other
- Advertise network resources
- Communicate with each other
- Cooperate with each other to form secure peer groups

The JXTA P2P platform enables application developers to build and deploy interoperable services and content, and is designed to help spring-board the peer-to-peer revolution.

Why JXTA?

JXTA has a set of objectives that are similar to, but distinct from current peer-to-peer systems. Most peer-to-peer systems are built for delivering a single type of network service (Napster for music file sharing, Gnutella for generic file sharing, AIM for instant messaging). Given the diverse characteristics of network services, and given the lack of a common underlying infrastructure, each vendor has created incompatible technologies that isolate their users from other P2P communities.

JXTA intents is to address this problem by providing a simple and generic P2P platform to host generic network services:

- JXTA uses a small number of protocols. Each is easy to implement and integrate into P2P services and applications. Thus service offerings from one vendor can be used transparently by the user community of another vendor's system.
- JXTA is independent of programming languages, so that it can be implemented in C/C++, the Java™ programming language, Perl, or other languages. Heterogeneous devices with completely different software stacks can interoperate with the JXTA protocols.
- JXTA is independent of transport protocols. It can be implemented on top of TCP/IP, HTTP, Bluetooth, HomePNA, and many other protocols. This means that a system built on top of JXTA functions in the same fashion when the system is expanded to a new networking environment or to a new class of devices, as long as there is a correct transport protocol handler for the new networking protocol.

The benefit of JXTA can be illustrated with a few examples.

Assume there is a P2P community offering a search capability for its members. In this community, one member can post a query and other members can hear and respond to the query. We can imagine that one member happens to be a Gnutella user and has cleverly implemented a feature so that whenever a query contains a question for a MP3 file, this member will look up the Gnutella directory and then respond to the query with information returned by the Gnutella system. As a result, a member with no knowledge of Gnutella can benefit because another member implemented a bridge to connect their P2P system to Gnutella. This type of bridging is very useful, but when the number of services is large, pair-wise bridging becomes more difficult and undesirable. JXTA aims to be the platform bridge that connects independent P2P systems together.

In another example, suppose one engineering group requires a sizable storage capability, but with redundancy to protect data from sudden loss. A common solution is to purchase a storage system with a large capacity and mirrored disks. Another engineering group down the hall later decides to purchase the same system. Both groups end up with a lot of extra capacity, and have to pay higher prices for the mirroring feature. With JXTA, the groups could create a system that would require that they only buy a simple storage system without mirroring. In this system, disks could discover each other automatically and form a storage peer group, mirroring data using their spare capacity.

As a third example, many devices (such as cell phones, pagers, wireless email devices, PDAs, and PCs) carry directory and calendar information. Currently, synchronization among them is tedious and difficult. Often, the PC becomes the central synchronization point, where every other device has to connect to the PC using a unique device driver for each device. With JXTA, all these devices could be made to interact with each other, without extra networking interfaces except those needed by the device themselves. JXTA would be the common layer of communication and data exchange.

JXTA Consists of Three Layers

The JXTA platform is divided in three layers.

- *Platform*. This layer encapsulates minimal and essential primitives that are common to P2P networking, including peers, peer groups, peer discovery, peer communication, peer monitoring, and associated security primitives. This layer is ideally shared by all P2P devices so that interoperability becomes possible. We sometimes refer to this layer as the core layer.
- *Services*. This layer includes network services that may not be absolutely necessary for a P2P network to operate ,but are common or desirable. Examples of network services include searching and indexing, directory, storage, file sharing, distributed file systems, resource aggregation and renting, protocol translation, authentication, and PKI services.
- *Applications*. This layer includes P2P instant messaging, entertainment content management and delivery, P2P email systems, distributed auction systems, and others. The boundary between services and applications is not rigid. An application to one customer can be viewed as a service to another customer.

JXTA's Technical Approach

Protocols

The common thread among JXTA peers is *protocols*, not APIs or software implementations. The JXTA protocols guarantee interoperability between compliant software components executing on potentially heterogeneous peer runtimes. JXTA is independent of programming languages.

The term *compliant* refers to a *single* protocol only. That is some peers may not implement all core protocols. Furthermore, some peers may only use a portion (client-side or server-side only) of a protocol.

The protocols defined in this document can be implemented over the Internet, a corporate intranet, a dynamic proximity network, in a home networking environment, or even within a single computer. JXTA is transport-protocol independent.

The size and complexity of the network peers supporting these protocols can range from a simple light switch to a complex, highly-available server.

Bindings

When JXTA protocols are implemented using a particular programming language and over a particular transport protocol, the implementation is an instance of a JXTA binding, where the JXTA protocols are bound to the language and the transport layer. Protocol and peer software implementation issues are defined in documents specific to the binding.

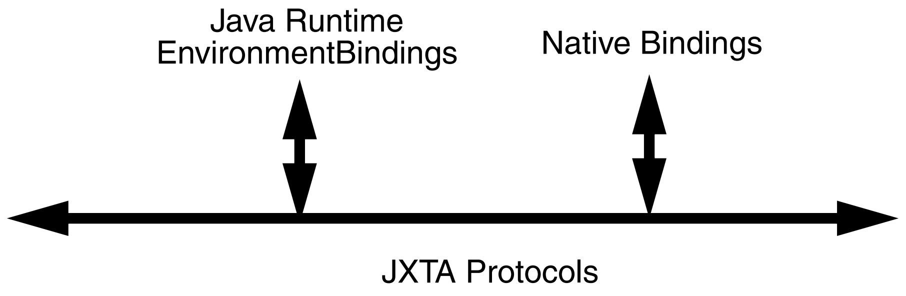


Figure 1: Protocols and bindings

A binding document describes how the protocols are bound to an underlying network transport (like TCP/IP or UDP/IP) or to a software platform such as UNIX® or the Java Runtime Environment.

Network Services

Peers cooperate and communicate with each other to locate and access *services*. Some services are well-known and are called JXTA *core services.* These include discovery and membership services. Other services are user-defined and provide application-dependent services such as content searching and indexing.

The JXTA core services are 100% de-centralized to enable pure P2P network computing. However, user-defined services can be implemented that may offer the ability to mix-in centralization as a means of increasing performance. For example:

- Efficient long-distance peer lookup and rendezvous using a *peer naming and discovery service*
- Simple low-cost information searching and indexing using a *content sharing service*
- Interoperability with existing centralized networking infrastructure and security authorities in corporate, public, private, or university networks using *administration services*

Advertisements

In JXTA, all network resources, such as peers, peer groups, and network services are represented by an *advertisement*. Advertisements are key pieces of information exchanged between peers. Advertisements are implemented using XML documents. Advertisements are used to represent all *JXTA resources* managed by the core platform including peers, peer groups, pipes, or services.

All advertisement documents are defined in XML and are therefore platform neutral. Each document may be converted to and from a platform-specific representation.

About This Document

This document specifies the JXTA architecture in terms of its

- Protocols
- Advertisements
- Core Services

Separate network protocol and software platform binding documents describe how the JXTA protocols are realized within a software platform such as the Java platform.

Network protocol bindings serve to ensure interoperability with existing content transfer protocols, network transports, routers, and firewalls.

This document contains a preface, an architecture overview, an advertisement discussion, and a detailed presentation of the "on-the-wire" protocols, including message formats.

The primary audience for this document is the JXTA developers, as well as the early platform adopters, and software developers building peer services and applications using JXTA technology.

The JXTA Architecture

JXTA combines network nodes called *peers,* into a simple and coherent peer-to-peer *network computing platform.*

The platform is build on very few principles:

- No single point of failure
- Asynchronous messaging
- Peers self-organize into secure peer groups
- Peers adapt to their network environment
- Contents are replicated and moved towards their consumers

The platform is completely decentralized, and with the addition of each new network peers, becomes more robust as it expands.

Peers

Network nodes of various kinds can join the platform by implementing one or more of the platform's protocols.

Each peer operates independently and asynchronously of any other peer, providing a degree of reliability and scalability not typically found in current distributed systems. Peers discover each other on the network in order to form transient relationships. Peers tend to be interchangeable and to interact mainly with few close neighbors.

Peer Protocols

All peer-to-peer protocols are embodied as XML messages sent between two peers. JXTA messages define the protocols used to discover and connect peers and peer groups, and to access network services offered by peers and peer groups.

The use of XML messages to define protocols allows many different kinds of peers to participate in a protocol. Each peer is free to implement the protocol in a manner best suited to its abilities and role. For example, not all peers are capable of supporting a Java runtime environment. The protocol definition does not require nor imply the use of a Java runtime environment.

Advertisements

Advertisements are JXTA's language neutral metadata structures. Each software platform binding describes how advertisements are converted to and from native data structures such as Java runtime objects or 'C' structures.

The complete list of the core JXTA advertisements is given in the advertisements chapter. The protocols chapter makes heavy reference to advertisements, so the reader should be familiar with advertisements before moving on to the protocol specification section.

Each protocol specification describes one or more request and response message pairs. Advertisements are by far the most common document exchanged in messages. The JXTA platform defines the following core advertisement types:

- Peer
- Peer Group
- Pipe
- Service
- Content
- Endpoint

User-defined advertisement subtypes (using XML schemas) may be formed from these basic types. The JXTA protocols and core software services however, operate *only* on the core advertisement types.

Messages

JXTA uses asynchronous XML *messages* as a basis for providing internet-scalable peer-to-peer communication.

Each peer's messaging layer delivers an ordered sequence of bytes from one peer to another peer in one atomic message unit.

Messages are sent between peer *endpoints*. A peer endpoint is a logical destination (embodied as a URI) on any networking transport capable of sending and receiving datagram-style messages. Endpoints are mapped into physical addresses by the messaging layer at runtime.

Note: JXTA does *not* assume that the networking transport is IP-based.

The messaging layer uses the transport specified by the URI to send and receive messages. Both reliable connection-based transports such as TCP/IP and unreliable connectionless transports like UDP/IP are supported. Other existing and emerging message transports like IRDA and Bluetooth are easily supported using the peer endpoint addressing scheme.

JXTA peer endpoint messages are datagrams that contain an *envelope and* a *stack* of protocol *headers* with *bodies*.

- The envelope contains a header, a message digest, source endpoint (optional), and destination endpoint.
- Each protocol header consists of a *tag***,** naming the protocol in use and a body length.
- Each protocol body is a variable number of bytes that is protocol tag dependent. Each protocol body contains one or more credentials used to identify the sender to the receiver.

When an unreliable networking transport is used, each message:

- May be delivered more than once to the same destination
- May not arrive at the destination
- May arrive in a different order than sent

High-level communication services layered upon the core messaging layer are responsible for message re-ordering, duplicate message removal, and for processing acknowledgement messages that indicate some previously sent message actually arrived at a peer. Regardless of transport, any message may:

- May be *unicasted* (point-to-point) between two peers
- May be *propagated* (like a multicast) to multiple peer members of a peer group. No multicast support in the underlying transport is required.

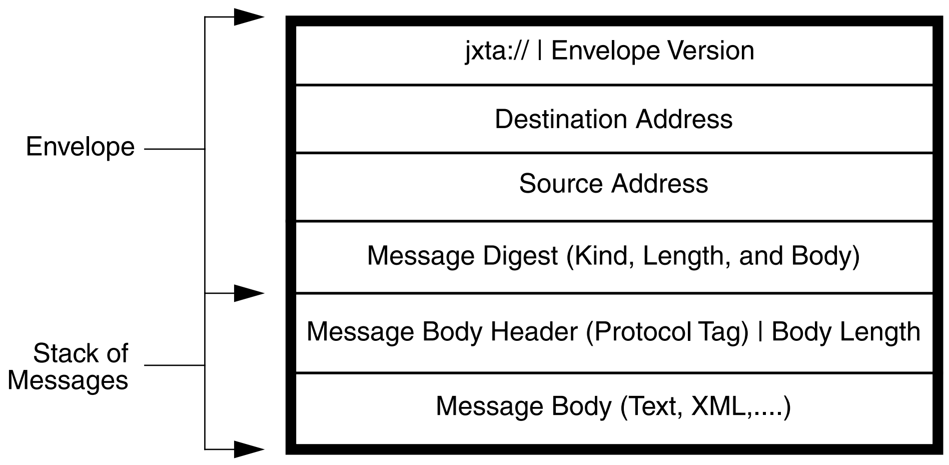
Message Credentials

A credential is a *key* that, when presented in a message body is used to identify a sender and their right to send the message to the specified endpoint. The credential is an opaque token that must be presented *each* time a message is sent.

The sending address placed in the message envelope is cross-checked with the sender's identity in the credential. Credentials are stored in the message body on a per-protocol tag basis. Each credential's implementation is specified as a plug-in configuration, which allows multiple authentication configurations to co-exist on the same network.

Message Definition

JXTA peer endpoint messages are defined using the following envelope:


Envelope
Stack of Messages
jxta:// | Envelope Version
Destination Address
Source Address
Message Digest (Kind, Length, and Body)
Message Body Header (Protocol Tag) | Body Length
Message Body (Text, XML,....)

The representation on the wire of the JXTA peer endpoint messages is as follows. It is important to point out that the message representation is not a well-formed and valid XML document. To improve efficiency, the intent is to NOT require XML at the low-level peer endpoint message layer.

```
<JxtaMessageVersion> version number “1.0”</JxtaMessageVersion>
<JxtaMessageDest> destination peer id </JxtaMessageDest>
<JxtaMessageSrc> source peer id </JxtaMessageSrc>
<JxtaMessageDigest> digest </JxtaMessageDigest>
<JxtaMessageTagName> tag </JxtaMessageTagName>
<JxtaMessageTagData> body </JxtaMessageTagData>
...........
<JxtaMessageTagName> tag </JxtaMessageTagName>
<JxtaMessageTagData> body </JxtaMessageTagData>
```

Pipes

Pipes are virtual communication channels used to send and receive messages between network services or applications over peer endpoints. Pipes are uni-directional, asynchronous, and stateless and provide a network abstraction over the peer endpoint transport.

Pipes connect one or more peer endpoints. At each endpoint, software to send or receive, and to manage associated pipe message queues is assumed, but *not* mandated. The pipe endpoints are referred to as input pipes and output pipes.

Pipe endpoints are dynamically bound to a peer at runtime, via the *Pipe Binding Protocol* (see Protocols chapter). Pipes provides the illusion of a virtual in and out mailbox that is independent of any single peer location. Network services and applications can communicate through pipes without knowing to which physical peer a pipe endpoint is bound.

When a message is sent into a pipe, the message is sent to all peer endpoints connected (listening) to the pipe. The set of currently connected pipe endpoints (input pipes) is obtained using the *Pipe Binding Protocol*.

A pipe offers two modes of communication:

- *Point-to-Point.* A point-to-point pipe connects *exactly two* pipe endpoints together, an input pipe that receives messages sent from the output pipe. No reply operation is supported. Additional information in the message payload (like a unique ID) is required to thread message sequences.

- *Propagate Pipe.* A propagate pipe connects multiple input and output pipe endpoints together. Messages flow into the input pipes from an output pipe (propagation source). A propagate message is sent to all listening input pipes. This process may actually create multiple copies of the message to. When peer groups map to underlying physical subnets in a one-to-one fashion, transport multicast be may considered as an implementation optimization for propagate pipes.

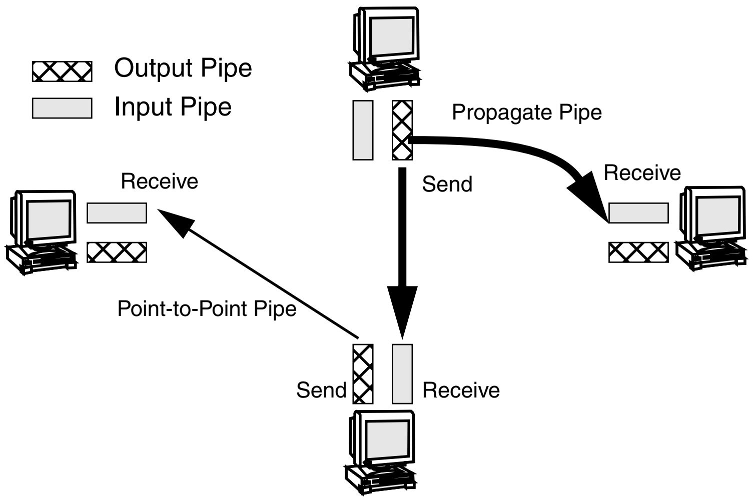


Figure 2: Message pipe modes

Network Services

JXTA peers cooperate and communicate to publish and access *services*. Each service has a unique ID and name that consists of a canonical *name string* and a series of *descriptive keywords* that uniquely identifies the service.

In order to actually access a service, a peer must locate an implementation suitable for the peer's runtime environment. For instance, multiple implementations of the same service might allow peers with Java runtime environments to use Java programming language implementations, and native peers to use native code implementations.

The JXTA platform recognizes two *levels* of services:

- Peer Services
- Peer Group Services

A peer service executes on a *single* peer network only. If that peer happens to fail, the service fails. This level of service reliability is acceptable for an embedded device, for example, providing a calendar and email client to a *single* user.

A peer group service, on the other hand, is composed of a *collection* of cooperating instances of the service running on multiple peers. If any one peer service fails, the collective peer group service is not affected, if the remaining peer services are healthy.

The JXTA platform defines a set of *core peer group services* used to build peer groups. The core peer group services provide the minimum services required to form and sustain a peer group (i.e. membership, pipe, monitoring, and discovery services). Other peer group services can be developed. For example, a resolver service could be implemented to find *active* (running on some peer) and *inactive* (not yet running) service instances.

Services can either be pre-installed into a peer or loaded from the network. The process of finding, downloading, and installing a service from the network is similar to performing a search on the Internet for a Web page, retrieving the page, and then installing the required plug-in.

Once a service is installed and activated, *pipes* may be used to communicate with it.

Service Advertisement

A service has a required name that also indicates the type or purpose of the service. An optional set of keywords further describes the service. The name and keyword elements are stored within a *service advertisement*.

The advertisement also contains other information needed to configure and instantiate a service or find the right type of implementation (e.g., Java runtime environment versus a native implementation).

Peer Groups

A peer group is a collection of cooperating peers that provide a common set of services. JXTA does not dictate when, where, or why a peer group should be created. The JXTA platform only describes how to create, manage, and discover peer groups.

Peers wishing to join a peer group must first locate a current member, and then make a request to join. The application to join is either rejected or accepted by the collective set of current members. A peer group *core membership* service enforces a vote or even elects designated group representative to accept or reject new membership applications.

JXTA recognizes *three common motivations* for creating or joining peer groups:

- To *create a secure cooperative environment*. Secure services can be provided to peers within a protected peer group. Peer groups form virtual secure regions. Their boundaries may or may not reflect any underlying physical network boundaries such as those imposed by routers and firewalls. The concept of a region virtualizes the notion of routers and firewalls, subdividing the network in a protected and self-organizing fashion without respect to actual physical network boundaries.

- To *create a restricted scope to ensure scalability.* Peer groups are formed primarily based upon the *proximity* of one peer to another peer. Proximity-based peer groups serve to subdivide the network into abstract *regions.* Regions serve as a placeholder for general communication and security configurations that deal with existing networking infrastructure, communication scopes and security requirements. Peer groups provide an efficient scoping mechanism to reduce traffic overload and search.

- *To create a controlled and self- administered environment*. Peer groups provide a self-organized structure that self-manage and can be locally managed.

Peer groups require:

- The ability to find nearby peers
- The ability to find named peers anywhere on the JXTA platform
- The ability to find named peer groups anywhere on the JXTA platform
- The ability to join and resign from a peer group
- The ability to establish pipes between peer group members
- The ability to find and exchange shared contents

Peer Group Core Services

The JXTA platform defines the following set of core services that enable the creation of peer groups. Not all core services need to be implemented by a peer group.

- *Discovery Service*. The discovery service is used to search for peers and peer groups. The search criteria may include a peer or peer group name (string).

- Access Service. The access service is used to validate, distribute, and authenticate a group member's credentials. The access service defines the type of credential in the message-based protocols used within the peer group.

- *Membership Service*. The membership service is used by the current members to reject or accept a new group membership application. (Used by current members during the login process.)

- *Pipe Service*. The pipe service is used to manage and create pipe connections between the different peer group members.

- *Resolver Service*. The resolver service is used to send a query string to peers to get information about a peer, peer group, a service, or a pipe.

Security

Peers, configurations, peer groups, and pipes form the backbone of the JXTA platform. Security in the JXTA platform relies on:

- *Credentials*. An opaque token providing an identity and a set of associated capabilities

- *Authenticators*. Code that receives messages and that requests a new credential that an existing credential be validated

All messages contain (at a minimum) a peer group credential that identifies the sender of the message as a full member in good standing.

- *Membership Credentials*. Membership credentials define a member's rights, privileges, and role within the group

- *Content Access and Sharing Credentials*. These credentials define a member's rights to the content stored within the group

Peer Group Content Sharing

One of the principal motivations for grouping peers together is to share content. An item of shared content can be a text file, a structured document (like a PDF or a XML file), or even active content like a network service.

Content is shared among group members, but *not* groups. No single item of content therefore, belongs to more than one group.

All items of content can be:

- *Identified (given a canonical name)*. Each item of content has a unique identifier also known as its *canonical* name. The composition of this name is a peer group UUID and another name computed, parsed, and maintained by peer group members. The content's name implementation within the peer group is *not mandated* by JXTA. The name could be a hash code, a URI, or any suitable means of uniquely identifying content within a peer group. The entire canonical content name is referred to as a *ContentID*.

128-bit UUID of Peer Group	Length of Remainder	Unique name within the group (A byte array)

- *Advertised*. Make item's existence known (available) to group members through the use of content advertisements.

Content

An instance of content is a copy of an item of content. Each content copy may reside on a different peer in the peer group. The only difference between the copies may be their encoding type (HTML or WML, for example). These copies will have the same ContentID, and could even exist on the same peer. The encoding metadata element is used to differentiate the two copies. Each copy has the *same* ContentID as well as a *similar set* of elements and attributes.

Making copies of content helps any single item of content be more available. For example, if an item has two instances residing on two different peers, only one of the peers needs to be available to respond to the content request.

The decision to copy an item of content is a configuration decision encapsulated in higher-level applications and services.

Active Content

Items of content that represent a network service are referred to as *active content*. These items have additional core elements above and beyond the basic elements used for identification and advertisement.

Active content is recognized by MIME content *type* and *subtype*. All JXTA active content has the same type. The subtype of active content is defined by network service providers and is *used to imply* the additional core elements belonging to active content documents.

The JXTA platform gives great latitude to service providers in this regard, yielding many service implementation possibilities.

JXTA Advertisements

An advertisement is an XML structured document that names, describes, and publishes the existence of a JXTA platform resource.

The JXTA platform defines the following core advertisement types:

- Peer
- Peer Group
- Pipe
- Service
- Content
- Endpoint
- User defined advertisement subtypes (using XML schemas) may be formed from these basic types. The JXTA protocols, configurations, and core software services, however, operate *only* on the core advertisements defined in this chapter.

This chapter describes all the required elements of each base advertisement type and some optional types. Subtypes may add an unlimited amount of extra, richer metadata.

XML

All JXTA advertisements are represented in XML. XML is a powerful means of representing data and metadata throughout a distributed system. XML provides universal (software-platform neutral) data because:

- XML is language independent
- XML is self-describing
- XML is strongly-typed
- XML ensures correct syntax

All XML advertisements are strongly typed and validated using XML schemas. Only valid XML documents that descend from the base XML advertisement types are accepted by peers supporting the various protocols requiring that advertisements be exchanged in messages.

The other powerful feature of XML is its ability to be translated into other encodings like HTML and WML. This feature supports peers that do not support XML to access advertised resources.

Advertisement Document Structure

Advertisements (like any XML document) are composed of a series of hierarchically arranged *elements*. Each element can contain its data or additional elements. An element can also have *attributes*. Attributes are name-value string pairs. An attribute is used to store meta-data, which helps to describe the data within the element.

Peer Advertisements

A *Peer Advertisement* describes peer information. The primary use of this document is to hold specific information about the peer, such as its name, peer id, registered services and available endpoints.

```
<?xml version="1.0" encoding="UTF-8"?>
<PeerAdvertisement>
   <Name> name of the peer</Name>
   <Keywords>search keywords </Keywords>
   <Pid> Peer Id </Pid>
   <Properties> peer properties </Properties>
   <Service> service advertisement</Service>
   ...........
   <Service> Service advertisement</Service>
   <Endpoint> endpoint Advertisement </Endpoint>
   ...........
   <Endpoint> endpoint Advertisement </Endpoint>
</PeerAdvertisement>
```

Element Name	Occurrence	Element Value Type
Name	Required	String
Keywords	0/1	String
Pid	Required	JxtaID
Properties	0/1	Properties
Service	*[a]	ServiceAdvertisement
Endpoint	+[b]	EndpointAdvertisement

a.'*' indicate 0 or more elements
b.'+' indicate 1 or more elements

Table 1: Peer Advertisement

Peer Group Advertisements

A *Peer Group Advertisement* describes the group specific information (name, id), the membership process, and the provided peer group services.

Once a peer joins a group, that peer may receive (depending again upon membership configuration) a full membership-level advertisement. The full membership advertisement, for example, might include the configuration (required of all members) to vote for new member approval.

```
<?xml version="1.0" encoding="UTF-8"?>
<PeerGroupAdvertisement>
   <Name> name of the peer group</Name>
   <Keywords>search keywords </Keywords>
   <Pid> Peer Id </Pid>
   <Gid> Peer group Id </Gid
   <Service> service advertisement</Service>
   ...........
   <Service> Service advertisement</Service>
</PeerGroupAdvertisement>
```

Element Name	Occurrence	Element Value Type
Name	Required	String
Keywords	0/1	String
Gid	Required	JxtaID
Pid	Required	JxtaID
Service	+	ServiceAdvertisement

Table 2: Peer Group Advertisement

Pipe Advertisements

A *Pipe Advertisement* describes an instance of a pipe communication channel. A pipe advertisement document is published and obtained using either the core discovery service or by embedding it within other advertisements such as the peer or peer group advertisement. Each pipe advertisement can include an optional symbolic name to name the pipe and a pipe type to indicate the type of the pipe (point-to-point, propagate, secure, etc).

```
<?xml version="1.0" encoding="UTF-8"?>
<PipeAdvertisement>
   <Name> name of the pipe</Name>
   <Id> Pipe Id </Id>
   <Type> Pipe Type </Type>
</PipeAdvertisement>
```

Element Name	Occurrence	Element Value Type
Name	0/1	String
Id	Required	JxtaID
Type	0/1	String

Table 3: Pipe Advertisement

Service Advertisements

A *Service Advertisement* describes an instance of a service running on a peer. All the built-in core services, for example, are made available to the JXTA platform by publishing a service advertisement. The service advertisement document is published and obtained using the PIP (Peer Information Protocol) for peer services or the PeerGroups Discovery Protocol for peer group services.

All service advertisements describe how to activate and/or use the service. The core peer group services (that each peer group must implement in order to respond to the messages described in the peer group protocol chapter) advertise their existence in this manner.

The access method for services is a schema of valid XML messages accepted by the service.

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceAdvertisement>
   <Name> name of the Service</Name>
   <Version> Version Id </Version>
   <Keywords>search keywords </Keywords>
   <Id> Service Id </Id>
   <Pipe> Pipe endpoint to access the service </Pipe>
   <Params> service configuration params </Params>
   <URI> service provider location</URI>
   <Provider> Service Provider</Provider>
   <AccessMethod> method </AccessMethod>
   ...........
   <AcessMethod> method </AccessMethod>
</ServiceAdvertisement>
```

Element Name	Occurrence	Element Value Type
Name	Required	String
Keywords	0/1	String
Id	Required	JxtaID
Version	Required	String
Pipe	0/1	PipeAdvertisement
Params	*	String
URI	0/1	URI
Provider	0/1	String
AccessMethod	+	Method

Table 4: Service Advertisement

Content Advertisements

A *Content Advertisement* describes a content document stored somewhere in a peer group.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentAdvertisement>
   <Mime-type> name of the pipe</Mime-type>
   <Size> Pipe Id </Size>
   <Encoding> Pipe Type </Encoding>
   <Id> Content Id</Id>
   <RefId> Content Id about </RefId>
   <Document> document </Document>
</ContentAdvertisement>
```

Element Name	Occurrence	Element Value Type
Mime-Type	Required	String
Size	Required	Long
Encoding	Required	String
Id	Required	JxtaID
RefId	0/1	JxtaID
Document	**Required**	Document

Table 5: Content Advertisement

- All items of content have an *ID*. Each ID is a unique identifier also known as its *canonical* name. The composition of this name is a peer group UUID and another name computed, parsed, and maintained by peer group members only. The content's name implementation within the peer group is *not mandated* by JXTA. The name could be a hash code, a URI, or any suitable means of uniquely identifying content within a peer group. The entire canonical content name is referred to as a *Content ID*.

128-bit UUID of Peer Group	Length of Remainder	Unique name within the group (A byte array)

- A *size* element that is the total size of the content in bytes is provided for all items, and is represented by an unsigned long (64-bits)

Each item of content may also contain the following "common" elements:

- A *MIME type* (encoding is deduced from type) of the in-line or referenced data.

- A *refId*. If the advertised content is another advertisement (based upon its type), this is the content ID of the referenced content. Otherwise, the element doesn't exist.

Endpoint Advertisements

An *Endpoint Advertisement* is used to describe each peer network interface or supported protocol on a peer. A peer can have many network interfaces that can be used as peer endpoints. Typically, there will be one peer endpoint for each configured network interface or protocol (IP, HTTP). The Endpoint Advertisement is a tag field in the peer advertisement (see *Peer Advertisement*) and describes the endpoints available on the member peer.

```
<?xml version="1.0" encoding="UTF-8"?>
<EndpointAdvertisement>
   <Name> name of the endpoint</Name>
   <Keywords> search string </Keywords>
   <Address> endpoint logical address </Address>
   <Transport> Transport </Transport>
</EndpointAdvertisement>
```

This advertisement document is published and obtained using either the core discovery service or by embedding it within other advertisements such as the peer advertisement. Each endpoint advertisement includes transport binding information about each network interface or transport protocol.

Endpoints are represented with a virtual endpoint address that embeds all necessary information to create a physical communication channel on the specific endpoint transport. For example, *tcp://123.124.20.20:1002* or *http://134.125.23.10:6002* are strings representing endpoint addresses.

Element Name	Occurrence	Element Value Type
Name	Required	String
Keywords	0/1	String
Address	Required	URI
Transport	Required	TransportAdvertisement

Table 6: Endpoint Advertisement.

JXTA Core Protocols

JXTA Protocol Overview

The JXTA platform is defined by six networking protocols. All six protocols needs not be implemented by every peer. A peer needs only implement the protocols that it uses.

- *Peer Discovery Protocol (PDP).* PDP allows a peer to discover other peers and peer group advertisements. This protocol is used to find members of *any* kind of peer group, presumably to request membership.
- *Peer Resolver Protocol (PRP).* PRP allows a peer to send simple search queries to another peer group member.
- *Peer Information Protocol (PIP).* PIP allows a peer to learn about other peers' *capabilities and status*.
- *Peer Membership Protocol (PMP).* PMP allows a peer *to join or leave peer groups*, and to manage membership configurations, rights, and responsibilities.
- *Pipe Binding Protocol (PBP).* PBP allows a peer to find the physical location of a pipe, and to bind a pipe endpoint to a physical peer endpoint transport.
- *Peer Endpoint Protocol (PEP).* PEP allows a peer to ask for peer routing information to route messages to another peer.

Some Definitions

A peer group can be viewed as an abstract region of the network, acting as a *virtual* sub-net. The concept of a peer group virtualizes the notion of routers and firewalls, subdividing the network in a self-organizing fashion without respect to actual physical network boundaries.

The term *region* is used throughout the remainder of this section to mean a peer group. Peer groups implicitly define a region scope that limits peer propagation requests.

By default, every peer is part of the *World Peer Group*. The world peer group is composed of all peers that run JXTA. The world peer group provides the minimum seed services for every peer to potentially find each other and to form new groups. The world peer group has a null membership authenticator.

The term *peer rendezvous* is used to designate a peer that elects itself to be a rendezvous point for discovering information about other peers, peer groups, services, and pipes.

Peer rendezvous typically cache information useful to other peer members to facilitate discovery of information in a peer group. Any or all members of a peer group can become rendezvous peers for a peer group. Each peer group may have different policies to authorize a peer to become a peer rendezvous.

Rendezvous peers provide an efficient mechanism for peers that are "far away" (firewalls, NAT, network gateways) from each other to find each other. Rendezvous peers are not required, but they make peer discovery more practical and efficient.

The term *peer router* is also used to designate a peer that crosses one or more networks and designates itself to be a router between the two networks. Any or all peer members *can become* routers. Peer groups may have different policies to authorize a peer to become a peer router for other peers.

Peer routers are used to route messages between different network transports (TCP/IP, Irda) or peers that are behind firewalls.

Peer Discovery Protocol

The *Peer Discovery Protocol* (PDP) enables a peer to find advertisements on other peers. PDP can be used to find any of the peer, peer group, or core advertisements. This protocol is the default discovery protocol for all user defined peer groups and the world peer group. Custom discovery services may or may not choose to leverage PDP. If a peer group does not have its own discovery service, the PDP will be used as a "lowest common denominator" method for probing peers for advertisements.

Rendezvous peers *may* keep a list of known peers and peer groups. This list may or may not be exhaustive or timely. A custom discovery service (if it knew that the rendezvous did kept a timely exhaustive list) could discover all peers by sending a *single message* to the rendezvous.

PDP supports finding peers with or without using a name. If a probing peer already knows the name of the peer to be located, a simple translation is requested that returns that peer's advertisement.

Once a peer is discovered, *ping*, *status, and capability* messages may be sent to its "main" endpoint(s) using PIP (See *Peer Information Protocol*). Peers may export more than one endpoint, but each peer should designate at least one "main" endpoint to handle the low-level housekeeping protocols such as PDP and PIP.

Finally, PDP can probe network peer groups looking for peers that belong to specified peer groups. This process is known as *screening*. Peers are screened for membership by presenting each candidate member with a peer group name (string matched with PeerGroupAdvertisement canonical name). Peers claiming to belong to this group respond, while others do not.

PDP is used to discover any type of core advertisements: peer advertisement, peer group advertisement, pipe advertisement, and service advertisement.

Messages to Discover Advertisements

- Get all known (reachable) advertisements within a region on the network. This list is not guaranteed to be exhaustive, and could even be empty. Named peers may also be located using PDP.
    - Message contains peer group credential of probing peer. Identifies probing peer to the message recipient.
- The destination address is :
    - any peer within a region (a propagate message)
    - a rendezvous peer (a unicast message)
- The response message returns one or more Peer Advertisements or Peer Group Advertisements that include "main" endpoint addresses converted to a string in the standard peer endpoint URI format (including network transport name).

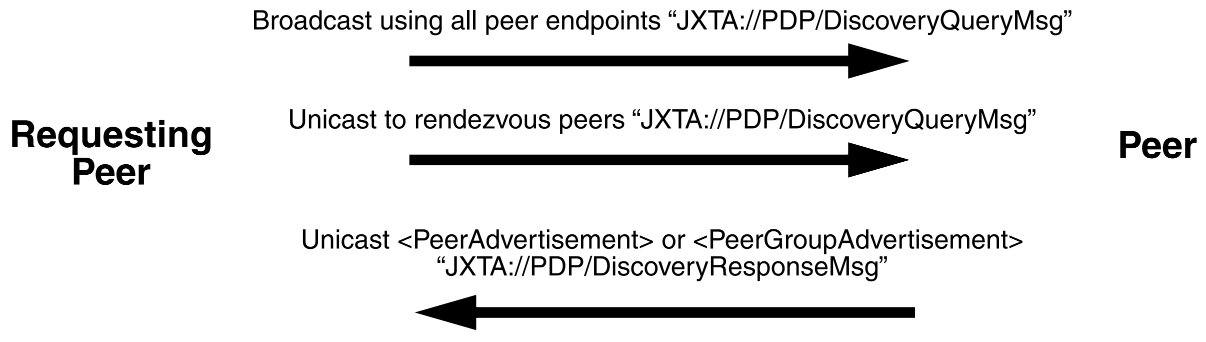

Broadcast using all peer endpoints "JXTA://PDP/DiscoveryQueryMsg"
Requesting Peer
Unicast to rendezvous peers "JXTA://PDP/DiscoveryQueryMsg"
Peer
Unicast <PeerAdvertisement> or <PeerGroupAdvertisement> "JXTA://PDP/DiscoveryResponseMsg"

Figure 3: Discovering advertisements.

Discovery Query Message

The discovery query message is used to send a discovery request to find peers or peer groups. The discovery query is sent as a query string (tag,value) form. A null query string can be sent to match any results. A threshold value is passed to indicate the maximum number of matches requested by a peer.

```
<?xml version="1.0" encoding="UTF-8"?>
<DiscoveryQueryMsg>
   <Credential> Credential </Credential>
   <Type> type of request (PEER, GROUP, ADV) </Type>
   <Threshold> requested number of responses </Threshold>
   <PeerAdv> peer advertisement of requestor </PeerAdv>
   <Query> query string (tag, value)</Query>
</DiscoveryQueryMsg>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
Type	Required	String
Threshold	Required	Int
PeerAdv	Required	PeerAdvertisement
Query	Required	String

Table 7: Discovery Query Message

Discovery Response Message

The Discovery response message is used to send a discovery response message to answer a discovery query message.

```
<?xml version="1.0" encoding="UTF-8"?>
<DiscoveryResponseMsg>
   <Credential> Credential </Credential>
   <Type> type of request (PEER, GROUP, ADV) </Type>
   <Count> number of responses </Count>
   <Adv> peer or peer group or pipe or service advertisement response
                  </Adv>
   ............
   <Adv> peer or peer group or pipe or service advertisement response
                  </Adv>
</DiscoveryResponseMsg>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
Type	Required	String
Count	Required	Int
Adv	+	Advertisement

Table 8: Discovery Response Message

Peer Resolver Protocol

The *Peer Resolver Protocol* (PRP) enables a peer service to send a generic query to another peer service. Each service of a peer group can register a handler in the core resolver service to process query requests. Resolver queries are de-multiplexed to each service. Each service can respond to the query via a resolver response message.

The PRP protocol enables each peer to send and receive generic queries to find or search for peer, peer group, pipe, or service-specific information such the state of a service, the state of a pipe endpoint.

Each resolver query has a unique service handler name to specify the receiving service and a query string to be resolved.

The Peer Resolver Protocol provides a generic mechanism for peer group services to send queries and receive responses. It removes the burden for registered message handlers by each service and set message tags to ensure the uniqueness of tags. The PRP protocol ensures that messages are sent to correct addresses and peer groups. It performs authentication and verification of credentials and drops rogue messages.

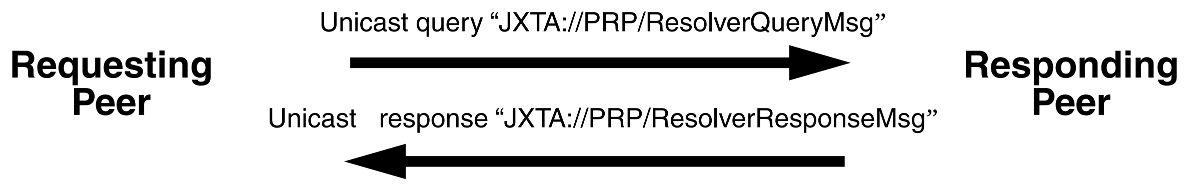

Unicast query "JXTA://PRP/ResolverQueryMsg"
Requesting Peer
Responding Peer
Unicast response "JXTA://PRP/ResolverResponseMsg"

Figure 4: Peer resolver protocol

Resolver Query Message

The resolver query message is used to send a resolver query request to another member of a peer group. The resolver query is sent as a query string to a specific service handler. Each query has a unique Id. The query string can be any string that will be interpreted by the targeted service handler.

```
<?xml version="1.0" encoding="UTF-8"?>
<ResolverQueryMsg>
   <Credential> Credential </Credential>
   <HandlerName> name of handler </HandlerName>
   <CredentialPolicyUri> uri to verify query credential
                  </CredentialPolicyUri>
   <QueryId> incremental query Id </QueryId>
   <Query> query string </Query>
   </ResolverQueryMsg>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
HandlerName	Required	String
CredentialPolicyURI	Required	URI
QueryId	Required	Int
Query	Required	String

Table 9: Resolver Query Message

Resolver Response Message

The resolver response message is used to send a resolver response message in response to a resolver query message.

```
<?xml version="1.0" encoding="UTF-8"?>
<ResolverResponseMsg>
   <Credential> Credential </Credential>
   <HandlerName> name of handler </HandlerName>
   <CredentialPolicyUri> uri to verify response credential
                  </CredentialPolicyUri>
   <QueryId> query Id </QueryId>
   <Response> response </Response>
</ResolverResponseMsg>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
HandlerName	Required	String
CredentialPolicyURI	Required	URI
QueryId	Required	Int
Response	Required	String

Table 10: Resolver Response Message

Peer Information Protocol

Once a peer is located, its capabilities and status may be of interest. The *Peer Information Protocol* (PIP) provides a set of messages to obtain this information.

Messages to Obtain Peer Status

To see if a peer is alive (responding to messages), send it a *ping* message.

- The destination address is the peer's "main" endpoint returned during discovery. The message contains the Group Membership Credential of requesting peer which identifies it to the message recipient. The message also contains an ID that is unique to the sender. This ID must be returned in the response.
- The response message returns an ID

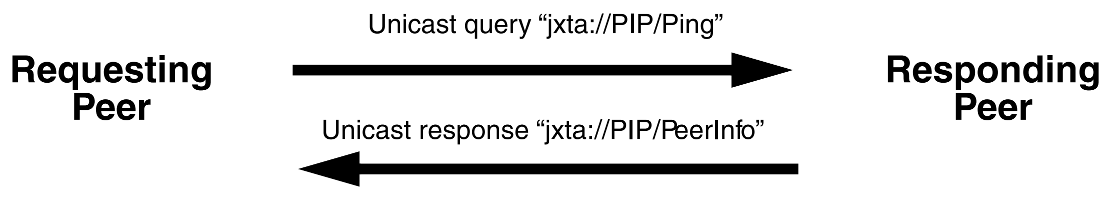

Unicast query "jxta://PIP/Ping"
Requesting Peer
Responding Peer
Unicast response "jxta://PIP/PeerInfo"

Figure 5: Peer Information Protocol

Messages to Get Peer Information

These messages get a list of named control "properties" exported by a peer's "main" endpoint. A property is a "knob" used to get information or configuration parameters from the peer. All properties are named (by a string), and are "read-only". Higher-level services may offer "read-write" capability to the same information, given proper security credentials.

Each property has a name and a value string. Read-write widgets allow the string value to be changed, while read-only widgets do not. PIP only gives read access.

The destination address is a peer's main endpoint returned in a discovery response message.

Ping Message

The ping message is sent to a peer to check if the peer is alive and to get information about the peer. The ping option defines the response type returned. A full response (peer advertisement) or a simple acknowledge response (alive and uptime) can be returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<Ping>
   <Credential> Credential </Credential>
   <SourcePid> Source Peer Id </SourcePid>
   <TargetPid> Target Peer Id </TargetPid>
   <Option> type of ping requested</Option>
</Ping>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
SourcePid	Required	JxtaID
TargetPid	Required	JxtaID
Option	Required	String

Table 11: Ping Message

PeerInfo Message

The peerrInfo response message is used to send a response to a ping message.

```
<?xml version="1.0" encoding="UTF-8"?>
<PeerInfo>
   <Credential> Credential </Credential>
   <SourcePid> Source Peer Id </SourcePid>
   <TargetPid> Target Peer Id </TargetPid>
   <Uptime> uptime</Uptime>
   <TimeStamp> timestamp </TimeStamp>
   <PeerAdv> Peer Advertisement </PeerAdv>
</PeerInfo>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
SourcePid	Required	JxtaID
TargetPid	Required	JxtaID
Uptime	Required	long
TimeStamp	Required	long
PeerAdv	Required	PeerAdvertisement

Table 12: PeerInfo Message

Peer Membership Protocol

The *Peer Membership Protocol* (PMP) allows a peer to:

- Obtain group membership requirements and application credentials
- Apply for membership and receive a membership credential along with a full group advertisement
- Update an existing membership or application credential

- Cancel a membership or an application credential

The first step to joining a peer group is to obtain a “form” listing the set of requirements asked of all group members. The form is a structured document (a peer group advertisement) that lists the Peer Group Membership service.

PMP defines several types of messages.

Membership Apply Message

jxta://PMP/Apply. This message is sent by a potential new group member to the group membership *application authenticator* (its endpoint is listed in the peer group advertisement of every member). A successful response from the group’s authenticator will include an application credential and a *group advertisement* that lists (at a minimum) the group’s membership service. The apply message contains:

- The current credential of the candidate group member
- The peer endpoint for the peer group membership authenticator to respond to with an ACK message

```
<?xml version="1.0" encoding="UTF-8"?>
<MembershipApply>
   <Credential> Credential of requestor </Credential>
   <SourcePid> Source pipe id</SourcePid>
   <Authenticator> Authenticator pipe adv</Authenticator>
</MembershipApply>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
SourcePid	Required	JxtaID
Authenticator	Required	PipeAdvertisement

Table 13: Membership Apply Message

Membership Join Message

jxta://PMP/Join. This message is sent by a peer to the peer group *membership authenticator* (its endpoint is listed in all peer group advertisements) to join a group. The peer must pass an application credential (from apply response ACK msg) for authentication purposes. A successful response from the group’s authenticator will include a full membership credential and a full group advertisement that lists (at a minimum) the group’s membership configurations requested of full members in good standing. The message contains two elements:

- An application credential of the applying peer (see ACK msg) that acts as the application form when joining
- The peer endpoint for the authenticator to respond to with an ACK message.

```
<?xml version="1.0" encoding="UTF-8"?>
<MembershipJoin>
   <Credential> Credential of requestor </Credential>
   <SourcePid> Source pipe Id </SourcePid>
   <Membersship> membership pipe Advertisement </Membership>
   <Identity> identity</Identity>
</MembershipJoin>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
SourcePid	Required	JxtaID
Membership	Required	PipeAdvertisement
Identity	Required	Identity

Table 14: Membership Join Message

Membership ACK Message

jxta://PMP/Ack. This message is an acknowledge message for both join and apply operations. It is sent back by the membership authenticator to indicate whether or nor the peer was granted application rights (peer is applying) or full membership (peer is joining) to the PeerGroup. The message contains the following fields:

- Credential (application or membership credential allocated to the peer by the peer group authenticator)
- A more complete peer group advertisement (providing access to further configurations). Not all configuration protocols are visible until the peer has been granted membership or application rights. Some configurations may need to be protected. Also, depending on the peer credential, the peer may not have access to all the configurations.)

```
<?xml version="1.0" encoding="UTF-8"?>
<MembershipAck>
   <Credential> Credential </Credential>
   <SourcePid> Source pipe Id </SourcePid>
   <Membersship> membership pipe adv</Membership>
   <PeerGroupAdv> peer group advertisement </PeerGroupAdv>
   <PeerGroupCredential> credential granted </PeerGroupCredential>
</MembershipAck>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
SourcePid	Required	JxtaID
Membership	Required	PipeAdvertisement
PeerGroupAdv	Required	PeerGroupAdvertisement
PeerGroupCredential	Required	Credential

Table 15: Membership ACK Message

Membership Renew Message

jxta://PMP/Renew. This message is sent by a peer to renew its credential (membership or application) access to the peer group. An ACK message is returned with a new credential and lease (if renew was “Accepted”). The renew message contains the following field:

- Credential (membership or application credential of the peer). The peer endpoint to send an ACK response message to.

```
<?xml version="1.0" encoding="UTF-8"?>
<MembershipRenew>
   <Credential> Credential </Credential>
   <SourcePid> Source pipe Id </SourcePid>
   <Membersship> membership pipe Adv</Membership>
</MembershipRenew>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
SourcePid	Required	JxtaID
Membership	Required	PipeAdvertisement

Table 16: Membership Renew Message

Membership Cancel Message

jxta://PMP/Cancel. This message is sent by a peer to cancel membership in or application rights in a peer group. The message contains the following fields:

- Credential (membership or application credential of the peer). The peer endpoint to send an ACK message. Successful ACK to a cancel contains only a response status of: “Accepted”.

```
<?xml version="1.0" encoding="UTF-8"?>
<MembershipCancel>
   <Credential> Credential </Credential>
   <SourcePid> Source pipe Id </SourcePid>
   <Membersship> membership pipe Adv </Membership>
</MembershipCancel>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
SourcePid	Required	JxtaID
Membership	Required	PipeAdvertisement

Table 17: Membership Cancel Message

Pipe Binding Protocol

The *Pipe Binding Protocol* (PBP) is used by peer group members to *bind* a pipe advertisement to a pipe endpoint. A pipe is conceptually a *virtual link* between two peer software components (services or applications).

The pipe virtual link (pathway) can be layered upon any number of physical network transport links such as TCP/IP. Each end of the pipe works to maintain the virtual link and to re-establish it, if necessary, by binding or finding the pipe’s currently bound endpoints.

Pipe Implementations and Transport Configurations

A pipe can be viewed as an abstract named message queue, supporting create, open/resolve (bind), close (unbind), delete, send, and receive operations. Actual pipe implementations may differ, but all compliant implementations use PBP to bind the pipe to a pipe endpoint. During the abstract “open” operation, a local peer *binds* a pipe endpoint to a pipe transport.

Each peer that “opens” a group pipe, makes an endpoint available (*binds*) to the pipe’s transport. Messages are only sent to one or more endpoints *bound* to the pipe. Peer members that have *not* opened the pipe do not receive, nor can they send, any messages in that pipe.

When peer software wants to accept incoming pipe messages, the receive operation removes a single message in the order it was received (not sent). A peek operation allows software to see if any message has arrived in the pipe's queue.

Pipe Messages

PBP defines several messages.

Pipe Binding Query Message

jxta://PBP/query: This message is sent by a peer pipe endpoint to find a pipe endpoint bound to the same pipe advertisement. The message contains the following fields:

```
<?xml version="1.0" encoding="UTF-8"?>
<PipeBindingQuery>
   <Credential> query credential </Credential>
   <Peer> optional tag. If present, it includes the URI of the only peer
                  that is supposed to answer that request.</Peer>
   <Cached> true if the reply can come from a cache </Cached>
   <PipeId> pipe id to be resolved </PipeId>
</PipeBindingQuery>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
Peer	0/1	URI
Cached	Required	boolean
PipeId	Required	JxtaID

Table 18: Pipe Binding Query Message

Pipe Binding Answer Message

jxta://PBP/answer. This response message is sent back to the requesting peer by each peer bound to the pipe. The message contains the following fields:

```
<?xml version="1.0" encoding="UTF-8"?>
<PipeBindingAnswer>
   <Credential> credential </Credential>
   <PipeId> pipe id resolved </PipeId>
   <Peer> peer URI where a corresponding InputPipe has been created
                  </Peer>
   <Found> true: the InputPipe does exist on the specified peer
                  (ACK)false: the InputPipe does not exist on the speci-
                  fied peer (NACK) </Found>
</PipeBindingAnswer>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
Peer	Required	URI
Found	Required	boolean
PipeId	Required	JxtaID

Table 19: Pipe Binding Answer Message

Endpoint Routing Protocol

The *Endpoint Routing Protocol* (ERP) is used by a peer router to send messages to another peer router to find the available routes a message may take to a destination. Two communicating peers may need to use router peers to route messages depending on the network topology. For instance, the two peers may be on different transports, or the peers may be separated by a firewall or a NAT router.

Routing information is represented as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<EndpointRouter>
   <Src> peer id of the source </Src>
   <Dest> peer id of the destination </Dest>
   <TTL> time to leave </TTL>
   <Gateway> ordered sequence of gateway </Gateway>
   ...................
   <Gateway> ordered sequence of gateway </Gateway>
</EndpointRouter>
```

Element Name	Occurrence	Element Value Type
Src	Required	JxtaID
Dest	Required	JxtaID
TTL	Required	int
Gateway	+	URI

Table 20: Endpoint Route

The ERP protocol is composed of a route request and a route answer from the router peer.

Peer routers will typically cache route information. Any peer can query a peer router for route information. Any peer in a peer group can become a peer router.

Route Query Request

This message is sent by a peer router to another peer router to request routing information. Routing information may or may not be cached. In some cases, the query can indicate to bypass the cache content of a router and search dynamically for a route.

```
<?xml version="1.0" encoding="UTF-8"?>
<EndpointRouterQuery>
   <Credential> credential </Credential>
   <Dest> peer id of the destination </Dest>
   <Cached> true: if the reply can be a cached replyfalse: if the reply
                must not come from a cache </Cached>
  </EndpointRouterQuery>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
Dest	Required	JxtaID
Cached	Required	boolean

Table 21: Endpoint Router Query

Route Answer Request

This message is sent by a router peer to a peer in response to a route information request.

```
<?xml version="1.0" encoding="UTF-8"?>
<EndpointRouterAnswer>
   <Credential> credential </Credential>
   <Dest> peer id of the destination </Dest>
   <RoutingPeer> URI of the router that knows a route to DestPeer
                </RoutingPeer>
   <RoutingPeerAdv> Advertisement of the routing peer </RoutingPeerAdv>
   <Gateway> ordered sequence of gateway </Gateway>
   ...................
   <Gateway> ordered sequence of gateway </Gateway>
</EndpointRouterAnswer>
```

Element Name	Occurrence	Element Value Type
Credential	Required	Credential
Dest	Required	JxtaID
RoutingPeer	Required	URI
RouterPeerAdv	Required	PeerAdvertisement
Gateway	+	URI

Table 22: Endpoint Router Answer

Draft 1.0 April 25, 2001