Jini™ Technology IP Interconnect Specification

Version 1.0 Standard
December 2004

IP

Jini™ Technology IP Interconnect Specification

IP.1 Introduction

It is assumed that the reader of this document has read and is familiar with the *Jini™ Technology Surrogate Architecture Specification.* The reader should also be familiar with basic internet protocol (IP) networking terms and protocols such as multicast, unicast, User Datagram Protocol (UDP), and Transmission Control Protocol (TCP).

A device[1] or surrogate host that uses the surrogate architecture for IP networks must comply with this specification, as well as with the *Jini™ Technology Surrogate Architecture Specification*.

An *interconnect* is defined in the *Jini™ Technology Surrogate Architecture Specification* as "the logical and physical connection between the surrogate host and a device." Although that specification does not provide details for any specific interconnect, it defines the set of requirements that must be met by an interconnect specification. These requirements include:

- Defining an interconnect-specific protocol
- Defining an interconnect-specific programming model

[1] As defined in the *Jini™ Technology Surrogate Architecture Specification*, in this context, the term *device* refers to either a *hardware* or *software* component.

IP.1.1 Overview

This specification meets the requirements of the *Jini™ Technology Surrogate Architecture Specification* by defining the following two components of the surrogate architecture for IP-capable interconnects:

- The IP interconnect protocol
- The IP-specific surrogate programming model

IP.1.1.1 The IP Interconnect Protocol

IP can be implemented for various physical media such as ethernet or wireless networks. The protocols defined by this specification are applicable to any physical layer that supports IP. The IP interconnect protocol specifies:

- *Discovery* - How a device and surrogate host discover each other

  Both the device and the surrogate host must have multicast capability to inform each other of their existence.
- *Retrieval* - How the surrogate is retrieved

  The IP interconnect protocol allows for the use of various data transfer mechanisms for retrieval of the surrogate, including File Transfer Protocol (FTP), Trivial File Transfer Protocol (TFTP), UDP, TCP, and Hypertext Transfer Protocol (HTTP).
- *Liveness* - How the device and the surrogate host communicate that the surrogate is functioning and the device for the surrogate is still reachable on the interconnect

  Liveness must be implemented by using a protocol between the surrogate and the device. The protocol used to communicate liveness is not mandated by this specification.

IP.1.1.2 The IP-specific Surrogate Programming Model

The IP-specific surrogate programming model defines IP-specific APIs that are available to the surrogate as well as additional security considerations.

IP.2 IP Interconnect Protocol

The IP interconnect protocol defines the way that a device and surrogate host discover each other and how the surrogate host retrieves the surrogate, runs it, and maintains its residence. This section describes the discovery protocols and the mechanisms for surrogate retrieval and liveness that make up the IP interconnect protocol.

IP.2.1 The Discovery Protocols

Discovery is the mechanism that the surrogate host and the device use to find each other on the interconnect. A surrogate host and a device must support the multicast host announcement and multicast host request protocols:

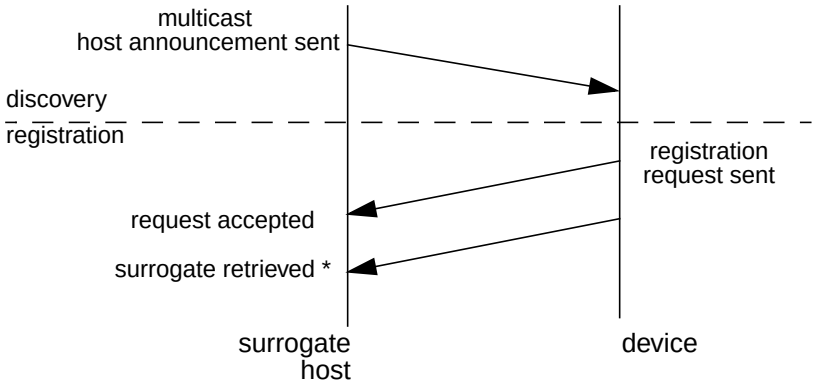
- The *multicast host announcement protocol* is used by the surrogate host to announce its availability over the interconnect.
- The *multicast host request protocol* is used by a device to discover a surrogate host at an unknown location.

The discovery protocols are provided so that the surrogate host and device do not have to be configured with specific location information. If a device can be configured with the location (IP address and port) of a specific surrogate host, it may bypass these discovery protocols and register directly with that host (see Section IP.2.4, “Surrogate Retrieval”). If this direct registration fails, the device should revert to the discovery protocols to find a suitable host. A device is not required to support configuration in any way.

IP.2.2 The Multicast Host Announcement Protocol

The multicast host announcement protocol is used by a surrogate host to announce its presence on the interconnect and to indicate that it is available to serve as a host

for a surrogate. A device needing a surrogate host listens for the multicast host announcement and, upon receiving the announcement, sends a registration request to the surrogate host.



* Although the line denoting surrogate retrieval originates from the device, in this and subsequent figures, the retrieval protocol allows for surrogate retrieval from a location other than the device.

FIGURE 2.1: ***Multicast host announcement protocol***

A surrogate host participating in the multicast host announcement protocol:

1. Creates listeners for UDP and TCP registration requests
2. Constructs a multicast host announcement containing the IP address and the ports used by the UDP and TCP registration request listener
3. Sends the multicast host announcement at intervals

A device participating in the multicast host announcement protocol:

1. Creates a listener for multicast host announcements
2. Makes a registration request to the host using the registration request protocol (see Section IP.2.4.1, “The Registration Request Protocol”) upon receiving a unicast host response

The surrogate host must send a series of no more then ten multicast host announcements when it is first initialized and then send an additional multicast host announcement at intervals. The length of the interval is not mandated, but 30 seconds is recommended and 120 seconds should be the maximum.

A surrogate host that is unable to accept registration requests must cease sending multicast host announcements. If the host is later able to accept registration requests, the host must resume sending multicast host announcements.

A device might receive more than one multicast host announcement from the same surrogate host. It might also receive a multicast host announcement from more than one surrogate host.

A device listens for multicast host announcements when it needs to locate a surrogate host. While a device has a relationship with an active surrogate, it may stop listening for multicast host announcements, but it must resume listening if it loses its connection with the surrogate.

IP.2.2.1 The Multicast Host Announcement

The *multicast host announcement* is a multicast UDP message. The message body must not exceed 512 bytes. The format of this message is:

```
short       message type
short       major protocol version
short       minor protocol version
short       length of host address
byte[]      host address
short       TCP registration port
short       UDP registration port
```

The fields have the following purposes:

- The `message type` field provides a way to distinguish between the types of discovery protocol messages. For the multicast host announcement message, the value of this field must be 1. This field is written as a two-byte integer in network byte order.
- The `major protocol version` field provides a way to distinguish between possible future changes to the protocol. For the current version of the protocol, the value of this field must be 1. This field is written as a two-byte integer in network byte order.
- The `minor protocol version` field provides a way to distinguish between possible future compatible extensions to the protocol. For a given major protocol version and minor protocol version *n*, compatible extensions will be indicated by a minor protocol version greater than *n*. For the current version of the protocol, the value of this field is 1. This field is written as a two-byte integer in network byte order.

- The `length of host address` field specifies the size in bytes of the `host address` field. This field is written as a two-byte unsigned integer in network byte order.
- The `host address` field contains the address of the surrogate host used by the device to make a registration request. The address is an IP address in dot-delimited notation or in an IPv6 address notation as defined by RFC2373. This field is written as 7-bit graphic printable ASCII characters. The length of this byte array is specified by the value of the `length of host address` field.
- The `TCP registration port` field contains the port on which the surrogate host is listening for TCP registration requests. This field is written as a two-byte unsigned integer in network byte order.
- The `UDP registration port` field contains the port on which the surrogate host is listening for UDP registration requests. This field is written as a two-byte unsigned integer in network byte order.

IP.2.3 The Multicast Host Request Protocol

The multicast host request protocol is used by a device to locate a surrogate host. A device that needs a surrogate host sends a multicast host request. The surrogate host, which is listening on the multicast port for the request, responds to the device with a unicast host response. The device uses the address in the unicast host response to generate a registration request for that host to serve as host for its surrogate. If the surrogate host accepts the request, the surrogate is retrieved.

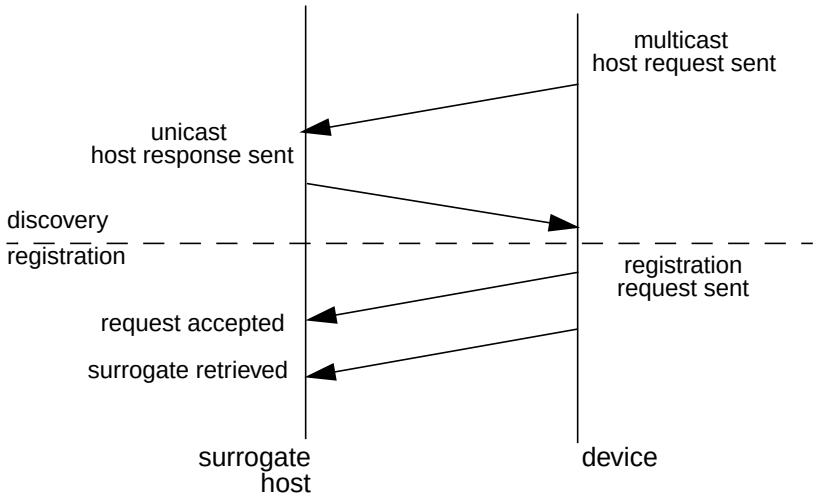


FIGURE 2.2: ***Multicast host request protocol***

A device participating in the multicast host request protocol:

1. Creates a unicast UDP listener to receive unicast host responses
2. Constructs a multicast host request that contains the IP address and port of the unicast host response listener
3. Sends a series of multicast host requests
4. Makes a registration request to the host using the registration request protocol (see Section IP.2.4.1, “The Registration Request Protocol”) upon receiving a unicast host response

A surrogate host participating in the multicast host request protocol:

1. Creates listeners for UDP and TCP registration requests
2. Creates a listener for multicast host requests on the host request multicast group
3. Constructs a unicast host response containing the IP address and the ports used by the UDP and TCP registration request listeners upon receiving a multicast host request

4. Sends the unicast host response to the device using the IP address and port contained in the multicast host request

When a device enters the interconnect, it should pause a random amount of time (up to 15 seconds is reasonable). This delay reduces the likelihood of a packet storm occurring if power is restored to an interconnect segment that houses a large number of devices. After pausing, the device should send no more than ten multicast host requests at periodic intervals. The interval at which requests are sent is not specified, though an interval of five seconds is recommended.

A device might receive more than one unicast host response from the same surrogate host. It might also receive a unicast host response from more than one surrogate host.

A surrogate host that is unable to accept registration requests should stop listening for multicast host requests. If the host stops listening for registration requests and is later able to accept registration requests, the host must resume listening for multicast host requests.

Once a device has a relationship with an active surrogate, it should stop sending multicast host requests. Should the device lose connectivity with the surrogate and still require a surrogate host, it may restart the multicast host request protocol.

IP.2.3.1 The Multicast Host Request

The *multicast host request* is a multicast UDP message. The message body must not exceed 512 bytes. The format of this message is:

```
short       message type
short       major protocol version
short       minor protocol version
short       length of device address
byte[]      device address
short       response port
```

The fields have the following purposes:

- The `message type` field provides a way to distinguish between the types of discovery protocol messages. For the multicast host request message, the value of this field must be 2. This field is written as a two-byte integer in network byte order.
- The `major protocol version` field provides a way to distinguish between possible future changes to the protocol. For the current version of the proto-

col, the value of this field must be 1. This field is written as a two-byte integer in network byte order.

- The `minor protocol version` field provides a way to distinguish between possible future compatible extensions to the protocol. For a given major protocol version and minor protocol version *n*, compatible extensions will be indicated by a minor protocol version greater than *n*. For the current version of the protocol, the value of this field is 1. This field is written as a two-byte integer in network byte order.
- The `length of device address` field specifies the size, in bytes, of the `device address` field. This field is written as a two-byte unsigned integer in network byte order.
- The `device address` field contains the address of the device to which the surrogate host sends a unicast host response. The address is an IP address in dot-delimited notation or in an IPv6 address notation as defined by RFC2373. This field is written as 7-bit graphic printable ASCII characters. The length of this byte array is specified by the value of the `length of device address` field.
- The `response port` field is the UDP port to which unicast host responses are sent. This field is written as a two-byte unsigned integer in network byte order.

If an error occurs during the processing of the multicast host request, the message is discarded.

IP.2.3.2 The Unicast Host Response

The *unicast host response* is a unicast UDP message that is sent by the surrogate host in response to a multicast host request. The message body must not exceed 512 bytes. The format of this message is:

```
short       message type
short       major protocol version
short       minor protocol version
short       length of host address
byte[]      host address
short       TCP registration port
short       UDP registration port
```

The fields have the following purposes:

- The `message type` field provides a way to distinguish between the types of discovery protocol messages. For the unicast host response message, the value of this field must be 3. This field is written as a two-byte integer in network byte order.
- The `major protocol version` field provides a way to distinguish between possible future changes to the protocol. For the current version of the protocol, the value of this field must be 1. This field is written as a two-byte integer in network byte order.
- The `minor protocol version` field provides a way to distinguish between possible future compatible extensions to the protocol. For a given major protocol version and minor protocol version *n*, compatible extensions will be indicated by a minor protocol version greater than *n*. For the current version of the protocol, the value of this field is 1. This field is written as a two-byte integer in network byte order.
- The `length of host address` field specifies the size, in bytes, of the `host address` field. This field is written as a two-byte unsigned integer in network byte order.
- The `host address` field contains the address of the surrogate host used by the device to make a registration request. The address is an IP address in dot-delimited notation or in an IPv6 address notation as defined by RFC2373. This field is written as 7-bit graphic printable ASCII characters. The length of this byte array is specified by the value of the `length of host address` field.
- The `TCP registration port` field contains the port on which the surrogate host listens for TCP registration requests. This field is written as a two-byte unsigned integer in network byte order.
- The `UDP registration port` field contains the port on which the surrogate host listens for UDP registration requests. This field is written as a two-byte unsigned integer in network byte order.

IP.2.4 Surrogate Retrieval

Retrieval of the surrogate is accomplished using the *registration request protocol*. The registration request protocol defines the messages and mechanism used by the device to request that a surrogate host load the device's surrogate JAR file. The device makes this request by sending a registration message to the surrogate host. The registration message contains either the surrogate itself, or the location of the

surrogate along with the protocol to use to retrieve the surrogate. The registration may also contain device-specific initialization data.

IP.2.4.1 The Registration Request Protocol

Once a device has obtained the IP address and port of the surrogate host, either through one of the discovery protocols or from configuration information, the device may send a registration request to that host in one of two ways. The device may send the request as a unicast UDP message or it may open a TCP connection to the surrogate host and send the message to the host via that connection. In either case, the format of the message is the same.

A device making a registration request:

1. Constructs a registration request
2. If the registration request is to be sent as a UDP packet, sends the registration request to the host using the IP address and UDP listener port of the surrogate host

   If the registration request is to be sent over a TCP socket, opens a TCP socket on the IP address and TCP listener port of the surrogate host and writes the registration request to that socket
3. Waits for confirmation from the device's surrogate that the surrogate has been activated by the host. The details of this step are specific to the implementations of the device and its surrogate and are not defined in this specification.

A surrogate host accepting a registration request:

1. Creates a listener for UDP registration requests
2. Creates a listener for TCP connections

The surrogate host may use different port numbers for the UDP and TCP listeners. These port numbers, along with the surrogate host's IP address, are provided to the device in the multicast host announcement and unicast host response (see Section IP.2.2.1, "The Multicast Host Announcement" and Section IP.2.3.2, "The Unicast Host Response").

A device attempting to register with a surrogate host should wait some amount of time for its surrogate to be activated before attempting to register again. The amount of time to wait is not defined by this specification. If the surrogate host remains unresponsive, the device should attempt to discover another host.

A device may attempt to register with more then one surrogate host. It is the responsibility of the surrogate and the device to coordinate multiple registrations.

If a surrogate host receives more than one registration request from the same device, it must attempt to honor all of the registrations. It is the responsibility of the surrogate and the device to determine if duplicate registrations have been made, and to then deactivate any unnecessary surrogates.

If a registration request is made through a TCP socket connection, the surrogate host may close that connection at any time after the registration request has been received by the surrogate host.

A surrogate host that is unable to accept registrations should stop listening for registration requests (both UDP and TCP). If the surrogate host stops listening for registration requests and is later able to accept registrations, it must resume listening for registration requests.

IP.2.4.2 The Registration Request

The registration request is sent by a device to a surrogate host. The format of the message is the same whether the registration is sent via a UDP message or through a TCP connection. The registration request has the following format:

```
short       major protocol version
short       minor protocol version
short       length of surrogate URL
byte[]      surrogate URL
int         length of initialization data
byte[]      initialization data
int         length of surrogate
byte[]      surrogate
```

The fields have the following purposes:

- The `major protocol version` field provides a way to distinguish between possible future changes to the protocol. For the current version of the protocol, the value of this field must be 1. This field is written as a two-byte integer in network byte order.
- The `minor protocol version` field provides a way to distinguish between possible future compatible extensions to the protocol. For a given major protocol version and minor protocol version *n*, compatible extensions will be indicated by a minor protocol version greater than *n*. For the current version of the protocol, the value of this field is 1. This field is written as a two-byte integer in network byte order.

- The `length of surrogate URL` field specifies the size, in bytes, of the `surrogate URL` field. This field is written as a two-byte unsigned integer in network byte order.
- The `surrogate URL` field specifies the location of the surrogate as well as the protocol to use to retrieve the surrogate JAR file. The URL must conform to the US-ASCII character encoding standard as defined by RFC 1738. The length of this byte array is specified by the value of the `length of surrogate URL` field.
- The `length of the initialization data` field specifies the size, in bytes, of the `initialization data` field. If no initialization data is to be provided, the value is 0. This field is written as a four-byte unsigned integer in network byte order.
- The `initialization data` field contains device-private data that is provided to the surrogate during activation. The contents of this field are not interpreted by the surrogate host. This field is written as a sequence of zero or more bytes. The length of this byte array is specified by the value of the `length of initialization data` field.
- The `length of surrogate` field specifies the size, in bytes, of the surrogate field. This field is written as a four-byte unsigned integer in network byte order.
- The `surrogate` field is a byte array that the surrogate host interprets as a JAR file containing the device's surrogate. The contents of the JAR file are described in the *Jini™ Technology Surrogate Architecture Specification*. This field is written as a sequence of zero or more bytes. The length of this byte array is specified by the value of the `length of surrogate` field.

If the `length of surrogate URL` field and the `length of surrogate` field are both zero or both non-zero, the message is invalid and the registration is discarded by the host.

The protocols that can be specified in the `surrogate URL` field are HTTP, FTP, and TFTP. A surrogate host must support all of these protocols.

Upon receipt of a registration request, the surrogate host takes the following steps:

1. If the surrogate URL is present in the registration request (meaning the `length of surrogate URL` field is non-zero), the surrogate host uses this URL to retrieve the surrogate JAR file.

If the surrogate URL is not present (length of `surrogate URL` field is zero), the surrogate host attempts to extract the surrogate JAR file from the registration message.

2. If initialization data is present (`length of initialization data` field is non-zero) the surrogate host retrieves this data and provides it to the surrogate through the `getInitializationData` method of the `IPInterconnectContext` interface, as described in Section IP.3.1.1, "The `IPInterconnectContext` Interface".
3. The surrogate host processes the surrogate JAR file and activates the surrogate as described in the *Jini™ Technology Surrogate Architecture Specification.*

If an error occurs during the processing of the registration request, the message is discarded. If the discarded registration request was made through a TCP connection, the surrogate host closes that connection.

The maximum size of a registration request that is sent as a UDP packet is 65,636 bytes minus the size of the UDP header and the IP header.

IP.2.5 Liveness

There are two components of liveness. First, the surrogate host needs to know if the device for which it is hosting a surrogate is still reachable on the interconnect. Second, the device needs to know if its surrogate is still resident and active in the surrogate host.

IP.2.5.1 Device Presence

The surrogate must continually test for the device's presence on the interconnect. The protocol used to determine the device's presence is specific to the implementation of the surrogate and the device. If the device is no longer present on the interconnect, has become non-responsive, or has sustained any other failure, the surrogate must be deactivated. If the device cannot be reached after some appropriate amount of time, the surrogate must deactivate itself by invoking the `cancelActivation` method of `net.jini.surrogate.HostContext`.

It is recommended that the surrogate use the keep-alive mechanism provided through the `KeepAliveManagement` interface (see Section IP.3.1.1, "The `IPInterconnectContext` Interface") for performing the task of testing for the device's presence.

IP.2.5.2 Surrogate Residence

It is the surrogate's responsibility to keep the device informed that it is still hosted by the surrogate host. The protocol used for informing the device is specific to the implementation of the surrogate and device. It is recommended that the residence protocol *not* require the surrogate to send a message indicating that it is about to be removed from the surrogate host. A time-based (lease) solution might better address the possibility of failure of the surrogate, surrogate host, or interconnect. The keep-alive mechanism (see Section IP.3.1.1, "The `IPInterconnectContext` Interface") can aid in the implementation of a time-based solution.

IP.2.6 Limiting the Scope of the Multicast

An entity should restrict the scope of any multicast messages that it sends by setting the time-to-live (TTL) field of outgoing packets appropriately. The value of the TTL field is not mandated, but setting it to 2 is recommended.

IP.2.7 Address and Port Mappings for Discovery Messages

The port number for multicast host requests and multicast host announcements is 1925. For multicast host requests, the IP address of the multicast group over which the requests must travel is 224.0.1.173. Multicast host announcements must use the multicast group 224.0.1.174.

IP.2.8 IPv6 Addressing

The `host address` field in the host announcement and host response messages, the `device address` field in the host request message, and the `surrogate URL` field in the registration request can contain an IPv6 address. Because of hardware or software platform limitations, a surrogate host or device may not be able to process an IPv6 address; in this case the surrogate host or device may ignore any message or request containing an IPv6 address.

IP.3 IP-Specific Surrogate Programming Model

The IP-specific surrogate programming model extends the surrogate programming model by defining IP-specific surrogate APIs and security.

IP.3.1 IP-specific Surrogate APIs

The IP-specific surrogate APIs provide the surrogate with information that is specific to the device that registered the surrogate. These APIs are provided through the `context` parameter of the surrogate methods.

IP.3.1.1 The `IPInterconnectContext` Interface

The `IPInterconnectContext` interface provides methods that enable the surrogate to access interconnect-specific information. The `context` parameter passed to the surrogate's `activate` method must implement this interface.

```
package net.jini.surrogate.ip;

public interface IPInterconnectContext
             extends net.jini.surrogate.KeepAliveManagement {

       byte[] getInitializationData();

       java.net.InetAddress getAddress();
}
```

The Semantics

- The `getInitializationData` method returns the initialization data provided by the device in the registration request (see Section IP.2.4.2, "The Registration Request"). If no initialization data was provided by the device, `null` is returned.

- The `getAddress` method returns the IP address of the device that sent the registration request for the surrogate.

The state of the `context` object is undefined if the surrogate is deactivated.

The `IPInterconnectContext` interface extends the `net.jini.surrogate.KeepAliveManagement` interface. The `KeepAliveManagement` interface is described in the *Jini™ Technology Surrogate Architecture Specification*. The algorithm for determining the keep-alive period is dependent on the implementation of the surrogate host.

IP.3.2 Security

IP.3.2.1 Permissions

The surrogate must be granted permission to connect back to the device. If the implementation of the surrogate host does not otherwise grant broad connection permissions to the surrogate, it must grant specific connection permission to the address of the device. For example, if the device was at IP address `129.145.70.158`, the following permission would allow the surrogate to make a connection back to the device:

```
java.net.SocketPermission “129.145.70.158”, “connect,accept”
```