PUCC Basic Protocol – light profile

(Version 2.0 - March 31, 2009)

Peer-to-Peer Universal Computing Consortium (PUCC)

Table of Content

1. Introduction 6
2. Terminology 7
2.1. Definitions 7
2.1.1. PUCC architecture definitions 7
2.1.2. Definitions in this document 7
2.2. Abbreviations 7
3. References 9
4. Goals and Requirements 10
4.1. Goals 10
4.2. Requirements 10
4.2.1. Extensibility 10
4.2.2. Layered design 10
4.2.3. Protocol functions 10
4.2.4. Efficiency 10
4.2.5. Co-existence with (Mobile) Internet 10
4.2.6. Convergence with existing standards 10
5. Protocol Overview 12
5.1. Protocol Design 12
5.1.1. PUCC Core Protocol 12
5.1.2. Protocols over PUCC Core Protocol 12
5.2. Operation mode 13
5.2.1. Light PUCC architecture 13
5.2.2. Pure PUCC architecture 13
5.2.3. Hybrid PUCC architecture 13
5.3. Communication mode 14
5.3.1. Proactive communication mode 14
5.3.2. Reactive communication mode 14
5.4. Implementation Design 14
5.4.1. Framing 14
5.4.2. Encoding 15
5.4.3. Error Reporting 15
5.4.4. Asynchrony 15
5.4.5. Authentication 15

5.4.6. Message Signature 15
5.4.7. Encryption 15
5.4.8. Protocol Stack 15
5.5. ID definition 15
5.5.1. Node ID 15
5.5.2. Message ID 16
5.5.3. Community ID 16
5.5.4. User ID 16
5.5.5. Session ID 16
5.5.6. Protocol ID 16
6. PUCC Basic Protocol – light profile 17
6.1. PUCC Core Protocol 17
6.1.1. Core element 17
6.1.2. ComType element 17
6.1.3. MsgID element 17
6.1.4. ReplyID element 18
6.1.5. MsgType element 18
6.1.6. CommunityID element 19
6.1.7. Source element 19
6.1.8. Destination element 19
6.1.9. TraceRoute element 20
6.1.10. HopCount element 20
6.1.11. SessionID element 20
6.1.12. MsgBody element 21
6.2. PUCC Basic Communication Protocol 22
6.2.1. Hello Method 22
6.2.2. Bye method 26
6.2.3. Resource Information Exchange method 27
6.2.4. Resource Information 31
6.5. PUCC Control Message Protocol 35
6.5.1. Error Report method 35
6.5.2. Diagnose method 37
6.5.3. Lookfor method 40
7. Transport protocol bindings 46

7.1. TCP Transport ... 46
7.1.1. Frame ... 46
7.1.2. Connection Type ... 47
7.2. UDP/IP Transport ... 49
7.2.1. Frame ... 49
7.3. Bluetooth Transport ... 51
7.3.1. L2CAP ... 51
7.3.2. SPP ... 52
7.4. IEEE1394 Transport ... 54
7.4.1. Frame ... 54
7.4.2. Communication Method ... 54
7.4.3. Configuration ROM ... 54
7.4.4. Address Resolution ... 54
7.5. OBEX Transport ... 55
7.5.1. Frame ... 55
7.5.2. Connection Type ... 56
7.5.3. Transport layer address ... 56
7.5.4. OBEX Header ... 56
7.5.5. OBEX role ... 56
7.6. HTTP Transport ... 59
7.6.1. Frame ... 59
7.6.2. Connection Type ... 61
7.6.3. Mapping of PUCC connection with HTTP session ... 61
7.6.4. Transport layer address ... 67
7.6.5. HTTP version ... 67
7.6.6. HTTP status code ... 67
7.6.7. HTTP header ... 67
Appendix A: Version History ... 69
Appendix B: First Peer Discovery ... 70
Appendix C: NAT Transmission ... 71
A) What's UPnP? ... 71
B) IGD (InternetGatewayDevice) ... 72
C) Application of UPnP to PUCC Platform (Light PUCC network) ... 73
Appendix D: Use cases ... 74

A) Distributed Metadata Search Application 74
B) Mobile Push Service 74
C) Small Personal Network Platform 74
Appendix E: DTD for PUCC Basic Communication Protocol 75
A) DTD for Hello message 75
B) DTD for HelloResponse message 75
C) DTD for Bye message 76
D) DTD for ResourceInformationRequest message 76
E) DTD for ResourceInformationResponse message 77
F) DTD for ResourceInformationAdvertise message 79
Appendix I: DTD for PUCC Control Message Protocol 80
A) DTD for ErrorReport message 80
B) DTD for Diagnose message 81
C) DTD for DiagnoseResponse message 82
D) DTD for Lookfor message 83
E) DTD for LookforResponse message 83
Appendix J: Namespace Definitions 86
A) PUCC Core Protocol 86
B) PUCC Basic Communication Protocol 86
C) PUCC Control Message Protocol 86

1. Introduction

At the present day, various digital equipments are connected together using various communication protocols such as internet, mobile network and so on. In the field of personal area network and home network, various networks such as ECHONET, DLNA (UPnP) are developed and utilized. These technologies are highly effective for specific problem areas and have been getting the position as standard. However, focusing on each technology, there are also some problems for users, for example, captive of venders, no interoperability.

In these surroundings we expect that with the widespread use of mobile phones, digital cameras, printers and digital TVs, the technologies which enable these equipments to connect to global networks, to be seamlessly utilized from various locations, are strongly desired. The goals of PUCC are just such enabling technologies. For bring about breakthrough, PUCC aims to research and develop computing environment which provide interoperability of various networks and advanced services using the peer-to-peer communication technology without troublesome procedure of users.

This PUCC Basic Protocol – light profile specification provides the protocol stack structure for PUCC Basic Protocol – light profile and the protocol specification for providing light PUCC communication functions for PUCC applications.

2. Terminology

2.1. Definitions

2.1.1. PUCC architecture definitions

The following terms are defined in PUCC Architecture Specification.

- Control Node;

- Node;

- Node ID;

- Community;

- Community ID;

- Multicast Group;

- Multicast Group ID;

- Communication Mode;

- Communication Type.

2.1.2. Definitions in this document

- Message;

- Message ID;

- Application Protocol ID;

- User ID;

- Session ID;

2.2. Abbreviations

TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IP	Internet Protocol
XML	eXtensible Markup Language
UUID	Universal Unique Identifier
FQDN	Fully Qualified Domain Name
P2P	Peer-to-Peer
WAP	Wireless Application Protocol
NAT	Network Address Translation
MIME	Multipurpose Internet Mail Extensions
TLS	Transport Layer Security
SSL	Secure Socket Layer
URL	Unique Resource Locator

URN	Uniform Resource Names
URI	Universal Resource Identifier
DTD	Document Type Definition
L2CAP	Logical Link Control and Adaptation Protocol
OBEX	Object Exchange Protocol
HTTP	Hypertext Transfer Protocol
BEEP	Blocks Extensible Exchange Protocol

3. References

- PUCC Architecture Specification Version 1.0
- PUCC Basic Protocol Version 1.0
- T. Bray et al., "Extensible Markup Language (XML) 1.0 (Second Edition) ," W3C Recommendation, October 2000.
- R. Fielding et al., "Hypertext Transfer Protocol -- HTTP/1.1," RFC2616, June 1999.
- J. Postel, "Transmission Control Protocol," RFC793, September 1981.
- J. Postel, "User Datagram Protocol," RFC768, August 1980.
- J. Postel, "Internet Protocol," RFC791, September 1981.
- The Bluetooth Special Interest Group, "Bluetooth"
- T. Berners-Lee et al., "Uniform Resource Identifiers (URI): Generic Syntax," RFC2396, August 1998.
- Infrared Data Association, "OBEX(Object Exchange) Protocol".

4. Goals and Requirements

4.1. Goals

The goals of this document are:

- To define general, scalable, extensible peer-to-peer protocols for for various devices such as mobile phone, home appliance and digital equipments.
- To leverage existing standards where possible, especially existing and evolving Internet, home network, mobile network standards.

4.2. Requirements

4.2.1. Extensibility

Each file (parameter) for describing the communication message in peer-to-peer protocols should be defined independent of any transport protocol so that the defined peer-to-peer communication message could be encapsulated by other transport protocols easily.

4.2.2. Layered design

For providing a scalable and extensible application environment for each node and each protocol, layered design for the nodes themselves and the communication protocols between nodes is highly desirable. Protocol components should be defined independently.

4.2.3. Protocol functions

When designing peer-topper protocols, the following functions are typical of those that should be considered: Node discovery, node information exchange among nodes, and function negotiation among nodes.

4.2.4. Efficiency

Performance of peer-to-peer protocols should be optimized. To this end, various technologies such as route optimization, resource sharing, and resource caching should be considered when designing peer-to-peer protocols. Since wireless networks have specific characteristics such as latency, low bandwidth, and expensive data packets compared to wired networks, traffic on wireless networks should be minimized.

4.2.5. Co-existence with (Mobile) Internet

The peer-to-peer architecture should be realized over the existing (Mobile) Internet, home network, Internet. To this end, existing components of the (Mobile) Internet such as i-mode, WAP, firewall, and NAT. should be considered and utilized, when designing the Peer-to-Peer architecture. Since the current work is considered to be implemented based on the i-mode Mobile Internet, the i-mode gateway and NAT should be taken into account.

4.2.6. Convergence with existing standards

When designing the peer-to-peer protocols, the use of existing standards should be considered where applicable. Such standards may include but not be limited to: W3C standards such as SOAP, and IETF standards such as HTTP, TCP, and BEEP.

5. Protocol Overview

5.1. Protocol Design

This specification defines PUCC Basic Protocol – light profile exchanging PUCC application messages over the light PUCC network.

The PUCC Basic Protocol – light profile is independent of IP transport protocol. Consequently the PUCC Basic Protocol – light profile can be implemented on TCP, Bluetooth and IEEE1394 and so on.

The PUCC Basic Protocol – light profile is designed as a layered structure (Figure 1). Each layer provides a set of services and/or functions to other services and applications through well-defined interfaces. Each layer of the protocol stack is accessible to the layers above.

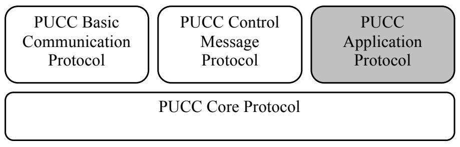

PUCC Basic Communication Protocol
PUCC Control Message Protocol
PUCC Application Protocol
PUCC Core Protocol

Figure 1. Protocol stack for PUCC Basic Protocol - light profile

The PUCC Basic Protocol - light profile is divided in two layers.

5.1.1. PUCC Core Protocol

This protocol provides the common peer-to-peer transport functions needed to establish or process a PUCC connection and/or session between two nodes on the light PUCC network. The functions provided by this protocol include designate communication mode, provide communication type, source information, destination information, traces route information and so on.

5.1.2. Protocols over PUCC Core Protocol

On top of the PUCC Core protocol, several PUCC communication and application protocols are defined. The protocols are functional independent of each other and each protocol provides a specific PUCC function. The functions defined on this protocol layer include initialize a PUCC connection and/or session, release an established connection and/or session, keep an established connection and/or session alive, and transfer specific PUCC application message.

5.1.2.1. PUCC Basic Communication Protocol

The PUCC Basic Communication Protocol is defined to realize the establishment and release of a PUCC connection and/or session. Additionally, this protocol has the function of exchanging resource information of a node such as names of its adjacent nodes and running PUCC applications in the node.

5.1.2.2. PUCC Control Message Protocol

The PUCC Control Message Protocol is defined to provide ancillary functions such as notification of a message forwarding error, keep-alive of PUCC connection and/or session, first PUCC node discovery and searching a PUCC node.

5.1.2.3. PUCC Application Protocols

The PUCC Application Protocols are defined to provide specific functions for each application. These protocols are designed specifically for each application.

5.2. Operation mode

Three PUCC architectures are defined for the PUCC network.

5.2.1. Light PUCC architecture

This is a minimum architecture for the PUCC network. There are only PUCC nodes in the Light PUCC architecture. Messages are sent from one PUCC node to another directly. Passing messages via several intermediary PUCC nodes is optional. A PUCC node as a client need not support relay functions. Routing information is discovered by broadcasting an inquiry message to the network.

In this architecture, a PUCC node needs to be accessible to PUCC Core Protocol, PUCC Basic Communication Protocol, and PUCC Control Message Protocol. Those protocols are PUCC Basic Protocol – light profile defined by this specification.

5.2.2. Pure PUCC architecture

There are only PUCC nodes in the pure PUCC architecture. Messages are sent from one PUCC node to another directly or by passing them via several intermediary PUCC nodes. Routing information is discovered by broadcasting an inquiry message to the network.

In this architecture, a PUCC node needs to be accessible to PUCC Core Protocol, PUCC Basic Communication Protocol, PUCC Multicast Communication Protocol, and PUCC Control Message Protocol included in PUCC Basic Protocol. To communicate with nodes of the light PUCC network, PUCC Basic Protocol – light profile is needed.

5.2.3. Hybrid PUCC architecture

The hybrid PUCC architecture consists of PUCC nodes and a control node. The control node provides a topology optimization function and security function as well responding to requests from PUCC nodes by returning routing information. The gateway node collects topology information on the pure PUCC network and passes it to the control node. A PUCC node in a hybrid PUCC network reports its own existence and adjacent nodes to the control node and can efficiently communicate with other PUCC nodes by using the routing information provided by the control node.

In this architecture, a PUCC node needs to be accessible to PUCC Core Protocol, PUCC Basic Communication Protocol, PUCC Basic Service Protocol, PUCC Multicast Communication Protocol, PUCC Multicast Service Protocol, and PUCC Control Message Protocol. Those protocols are PUCC Basic Protocol. The hybrid PUCC network can be connected to the pure PUCC network using a gateway node. The gateway node has a function to conbine the hybrid PUCC network and the pure PUCC network.

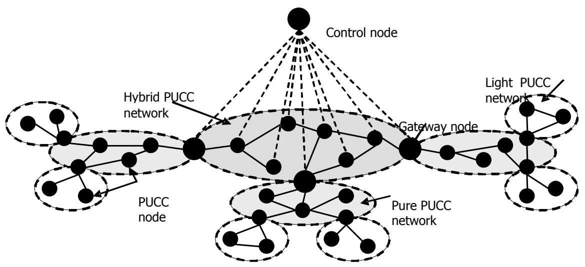


Figure 2. Structure of PUCC network

5.3. Communication mode

Two communication modes are defined for PUCC communications. They are proactive communication mode and reactive communication mode. The two communication modes are according to different communication needs.

5.3.1. Proactive communication mode

In the proactive communication mode, a node initiates communication to other nodes without any request.

5.3.2. Reactive communication mode

In the reactive communication mode, a node initiates communication only when it receives a communication request from another node.

5.4. Implementation Design

The PUCC Basic Protocol – light profile has a framing mechanism that permits simultaneous and independent exchanges of messages between nodes. Messages are arbitrary MIME content. In this specification, all messages are structured using XML.

The messages are defined using the following XML envelope:

```
<Core>
  Elements of parameter of   PUCC Core Protocol

  <MsgBody>
    Elements of parameters of

    PUCC Basic Communication Protocol
    PUCC Control Message Protocol
    PUCC Application Protocol
  </MsgBody>
</Core>
```

Figure 3. Message envelope using XML for light profile

Message blocks are distinguished from each other using the XML name space.

The concepts of PUCC Basic Protocol – light profile design include:

5.4.1. Framing

Message framing in the PUCC Basic Protocol – light profile is defined using the XML envelope. PUCC Basic

Protocol – light profile messages are encapsulated using <Core> start tag and ended by </Core>, and so can be distinguished from other messages.

5.4.2. Encoding

The PUCC Basic Protocol – light profile messages are encoded in text/xml MIME content and use UTF-8 character code set for describing protocol messages.

5.4.3. Error Reporting

Use network communication failure, XML format error, and protocol format error for each layer. Define error description in PUCC Control Message Protocol for each protocol.

5.4.4. Asynchrony

PUCC communication messages are distinguished from each other by using Message ID.

5.4.5. Authentication

Use TLS like protocol (for details refer to PUCC Security Architecture Specification).

5.4.6. Message Signature

We use XML signature (for details refer to PUCC Security Architecture Specification).

5.4.7. Encryption

We use XML Encryption (for details refer to PUCC Security Architecture Specification).

5.4.8. Protocol Stack

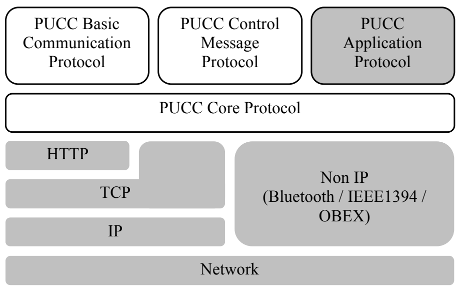

PUCC Basic Communication Protocol
PUCC Control Message Protocol
PUCC Application Protocol
PUCC Core Protocol
HTTP
TCP
IP
Non IP (Bluetooth / IEEE1394 / OBEX)
Network

Figure 4. Protocol stack for PUCC Basic Protocol – light profile (including transport and network layer)

5.5. ID definition

The PUCC Basic Protocol – light profile refers to PUCC nodes, PUCC messages, communities, users, sessions and protocols using the different kind of IDs. The PUCC Basic Protocol - light profile uses 6 types of IDs to distinguish the entities of a PUCC network.

5.5.1. Node ID

The Node IDs refer to PUCC nodes. A Node ID must be unique in a PUCC network. For example, UUID can be used as Node ID.

5.5.2. Message ID

The Message IDs refer to PUCC messages. A Message ID must be eternally unique in a PUCC network. For example, we can generate unique Message IDs using "[node provided sequence number].[current time(ISO 8601)]@[node ID]".

5.5.3. Community ID

The Community IDs refer to PUCC communities. A Community ID is globally unique. URN is recommended for the notation of Community IDs.

5.5.4. User ID

The User ID identifies the user who is using a PUCC session. A User ID shall be unique on a PUCC node.

5.5.5. Session ID

The Session IDs refer to the IDs that identify PUCC sessions. A PUCC session consists of a pair of user IDs; one setting up the PUCC session and the other the PUCC session is set up. セッション ID は、PUCC セッションを構築した PUCC ノード間で一意でなければならない。セッション ID は、PUCC セッションの Initiator 側ノードまたは Responder ノードで付与する。例えば、Hello メッセージのメッセージ ID をセッション ID に用い、Initiator 側ノードで付与することで ID の一意性を保つことができる。

5.5.6. Protocol ID

The Protocol IDs refer to PUCC Application Protocols. The Protocol ID is used for distinguish the PUCC Application Protocols. A Protocol ID is globally unique in the system. URI is recommended for the notation of Application IDs.

7A3260162A 2009/7/31 10:59
削除: The Session ID identifies a PUCC session. A PUCC Session consists of a combination of two user IDs: the one of the user who initiates a PUCC session and the one of the user to whom a PUCC session is initiated by the other user. The Session ID shall be unique on the relevant two PUCC nodes and the PUCC connection between those PUCC nodes. The Session ID is given by the PUCC node which initiates the session.

Osano 2008/5/26 14:34
書式変更: 箇条書きと段落番号

6. PUCC Basic Protocol – light profile

6.1. PUCC Core Protocol

The following are definitions of the fields (Parameter) in the Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Core	Following XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast","Broadcast"	-	-	1	Required	
MsgID	Message ID (String)	-	-	1	Required	
ReplyID	Message ID (String)	-	-	1	MsgType dependent	
MsgType	"Request","Response", "Advertise"	-	-	1	Required	
CommunityID	Community ID (String)	-	-	1	Optional	
Source	Node ID (String)	-	-	1	Required	
Destination	Route,Target element	-	-	1	ComType dependent	
Route	Route,Target element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID, (String)	-	-	1	Required	
TraceRoute	Route element	-	-	1	Optional	
Route	Route element	-	-	1	Required	
		Node	Node ID(String)	1	Required	
HopCount	Hop count (Integer)	-	-	1	Optional	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML document	-	-	1	Required	
		protocol	URI	1	Required	

Table 1: Fields of PUCC Core Protocol

6.1.1. Core element

The Core element is defined for encapsulating PUCC messages to distinguish them from other data. Every PUCC message should be structured under the Core element. This element has xmlns attributes which designate the namespace of the PUCC Core Protocol. Eleven elements are defined under the Core element.

6.1.2. ComType element

The ComType element designates the communication type that the message uses.

Two communication types are defined in this specification: unicast and broadcast. When the communication type is described as unicast, the message should be sent to one node indicated by node ID. When the communication type is described as broadcast, the message should be sent to all nodes except for a control node within the PUCC network. This field is essential.

6.1.3. MsgID element

The MsgID element is unique and distinguishes the message from other messages. In this specification, the message ID value is defined as "[node provided sequence number].[current time(ISO 8601)]@[node ID]". This field is essential in each PUCC protocol message envelope.

Ex) 12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352

6.1.4. ReplyID element

The Reply ID element is to designate the MsgID of the message replied. This field is essential in response message (MsgType="Response"). In other cases, it is ignored.

6.1.5. MsgType element

The MsgType element is to designate which PUCC message type is used by the current message. In this specification, there are three PUCC message types: advertise, request, and response. This field is essential in each PUCC protocol message envelope.

6.1.5.1. Sequence of Messages

6.1.5.1.1. Advertise message

This is a message that demands no response. A sequence consists of one advertise message named "Proactive Communication Mode".

6.1.5.1.2. Request message

This is a message that demands a response from the receiving nodes.

6.1.5.1.3. Response message

This is a message issued in response to a request message. A sequence consisting of a request message and a response message is named "Reactive Communication Mode".

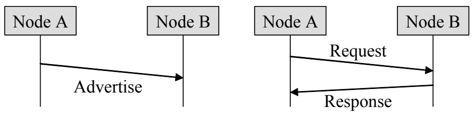


Figure 5. Sequences of Proactive (left) and Reactive (right)

6.1.5.2. Methods of Transmitting Response messages

There are two methods of transmitting response messages. The transmission method selected depends on the node's decision.

6.1.5.2.1. Direct response method

The direct response method is to transmit the response message directly to the source node with reference to the Source element. Even if the request message passed through several nodes, the response message is transmitted directly to the source node.

6.1.5.2.2. Chained response method

The chained response method is to transmit the response message along the path traversed by the request message with reference to the TraceRoute element.

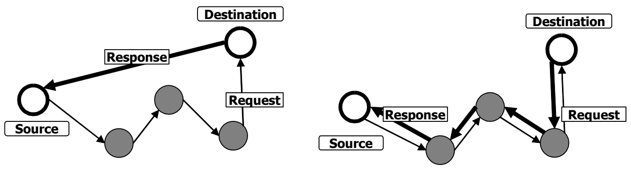


Figure 6. Direct Response (left) and Chained Response (right)

6.1.6. CommunityID element

The CommunityID element designates a community. All messages must contain this element. URN is recommended for the notation of CommunityID. If CommunityID element is empty or not specified, it is regarded as the default community. If a node receives a message with a different community ID, the node discriminates the message.

Ex) urn:pucc:community:DoCoMo

6.1.7. Source element

The Source element designates the Node ID of the node that sent a message. This element is essential when a PUCC node sends any kind of message.

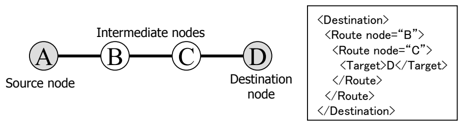
6.1.8. Destination element

The Destination element designates the destination of messages; the route taken to reach the destination is specified using the Node IDs of the nodes to be transited. When the communication type of the message is unicast, this field is essential. When the communication type is broadcast, this field should be dropped.

The Destination element does not include the Node ID of the source node. It is designed by the source node.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
Destination	Route,Target element			1	ComType dependent	
Route	Route,Target element			1	Optional	
		Node	Node ID	1	Required	
Target	Node ID (String)			1	Required	.

Table 2: Destination element



```
<Destination>
  <Route node="B">
    <Route node="C">
      <Target>D</Target>
    </Route>
  </Route>
</Destination>
```

Figure 7. A sample of a route map and a notation of Destination element

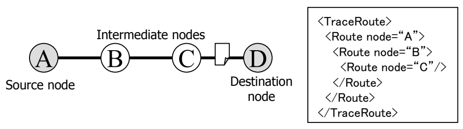
6.1.9. TraceRoute element

The TraceRoute element contains the route passed by a message. When this element is used (the source node designates TraceRoute), every node on the message route appends its own node ID information to this element. TraceRoute includes the Node IDs of source node and destination node. This element is essential for the following cases.

1. A message belongs to Resource Information Exchange method or Diagnose method
2. ComType element = “Broadcast” and MsgType element = “Request”

Element name		Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
TraceRoute		Route element	-	-	1	Optional	
	Route	Route element	-	-	1	Required	
			Node	Node ID（String）	1	Required	

Table 3: TraceRoute element



```
<TraceRoute>
  <Route node="A">
    <Route node="B">
      <Route node="C"/>
    </Route>
  </Route>
</TraceRoute>
```

Figure 8. A sample of a route map and a notation of TraceRoute element

6.1.10. HopCount element

The HopCount element designates the maximum number of nodes the message should pass through. When the message reaches a node, the hop counter is decreased by one, and when the hop count reaches zero, the sent message should be destroyed. This field is essential in Broadcast message (ComType=”Broadcast”) but not “flooding”. In other cases, it is ignored. If HopCount is not described in a Broadcast message, the value of HopCount is construed as infinity.

6.1.11. SessionID element

The Session ID element is an identifier, which identifies a session between PUCC nodes where per-user PUCC messages are sent and received. A PUCC session can be identified based on a combination of two user IDs; the one of the user who initiates a PUCC session and the one of the user to whom a PUCC session is initiated by the other user. The user who initiates the PUCC session, namely the user who sends a Hello message assigns the value which represents the above-mentioned combination to their Session ID element. The Session ID element of the PUCC messages transmitted in the same PUCC session shall be set to the same value.

The table below shows the necessity/non-necessity of setting the ID for the users who constitute a PUCC session, namely the user who initiates a session and the user to whom a session is initiated, the necessity/non-necessity of Session ID elements and their correlations. When the ID of the user who initiates a session or the user to whom a session is initiated is omitted, it shall be assumed that the user ID which indicates nonexistence of the user is set.

Table 4: Setting of User ID and Session ID

	ID of the user who initiates a session	ID of the user to whom a session is initiated	Session ID element
1	No	No	No
2	Yes	Yes	Yes
3	Yes	No	Yes
4	No	Yes	Yes

The number of sessions that can be concurrently established between the user who initiates a session and the user to whom a session is initiated is only one. Initiation of more than one session shall not be allowed. When the user who initiates a session sends a Hello message to the user to whom a session has already been established, the user to whom the session has been established returns a HelloResponse (Failure) message with the Reason element (the cause of failure) set to "SessionAlreadyEstablished" so as not to initiate another session.

Any SessionID element is not set for a Lookfor message or a LookforResponse message.

The user who initiates a session and the user to whom a session is initiated do not relay multi-hop unicast, multicast or broadcast messages.

6.1.12. MsgBody element

The MsgBody element is defined for encapsulating the upper PUCC protocols and application message fields to distinguish it from the PUCC Core Protocol. The MsgBody element should be the last child of the PUCC Core Protocol and the protocol attribute include protocol ID to distinguish PUCC Application Protocols encapsulate by the MsgBody element.

6.2. PUCC Basic Communication Protocol

The PUCC Basic Communication Protocols provide the basic communication services to the nodes on the PUCC network for communicating with each other. The PUCC Basic Communication Protocol includes

- Hello method

The Hello method is used to establish a PUCC connection between PUCC nodes and/or a PUCC session between PUCC users. The Hello method is a Reactive (Request/Response) type method and consists of a Hello message and a HelloResponse message.

- Bye method

The Bye method is used to inform other nodes before the node release a PUCC connection and/or a PUCC session to the network. The Bye method is a Proactive (Advertise) type method and consists of a Bye message.

- Resource Information Exchange method

The Resource Information Exchange method is used to exchange resource information on other nodes on the network. The Resource Information Exchange method combines Reactive and Proactive methods. The reactive type resource exchange method consists of a ResourceInformationRequest message and a ResourceInformationResponse Message. The proactive type resource exchange method consists of a ResourceInformationAdvertise message.

6.2.1. Hello Method

6.2.1.1. Hello message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Hello	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
InitiatorUserID	Initiator User ID (String)	-	-	1	Optional	
ResponderUserID	Responder User ID (String)	-	-	1	Optional	

Table 5: Fields of Hello message

The Hello message is mapped as per following table.

Element name	Element Value	Attribute name (if it has)	Attribute Value	Occurrence	Status	Description
Core	Following XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID (String)	-	-	1	Required	
ReplyID	Message ID (String)	-	-	1	unused	
MsgType	"Request"	-	-	1	Required	5
CommunityID	Community ID (String)	-	-	1	Optional	
Source	Node ID (String)	-	-	1	Required	
Destination	Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		-	-	-	unused	

Target	Node ID(String)	-	-	1	Required	
TraceRoute	-	-	-	-	unused	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 6: Mapping of Hello message to PUCC Core Protocol

The following is a sample of the Hello message without user ID and session ID.

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <MsgType>Request</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <Destination>
    <Target>874542ab-a5c6-4305-8745-f98e56f12547</Target>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <MsgBody protocol=” Namespace of PUCC Basic Communication Protocol” >
     <Hello xmlns=” Namespace of PUCC Basic Communication Protocol” />
  </MsgBody>
</Core>
```

Figure 9. Sample of Hello message without user ID and session ID

The following is a sample of Hello message with session ID and user ID.

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <MsgType>Request</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <Destination>
    <Target>874542ab-a5c6-4305-8745-f98e56f12547</Target>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <SessionID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</SessionID>
  <MsgBody protocol=” Namespace of PUCC Basic Communication Protocol” >
    <Hello xmlns=” Namespace of PUCC Basic Communication Protocol” >
      <InitiatorUserID>urn:pucc:user:A</InitiatorUserID>
      <ResponderUserID>urn:pucc:user:B</ResponderUserID>
    </Hello>
  </MsgBody>
</Core>
```

Figure 10. Sample of Hello message with user ID and session ID

6.2.1.1.1. Hello element

The Hello element is used to initialize a PUCC connection from one node to another node and/or a PUCC session from one user to another user on the PUCC network. Two elements are defined as fields (Parameters) in the Hello message.

6.2.1.1.2. InitiatorUserID element

The InitiatorUserID element sets the ID of the user who initiates a per-user session between PUCC nodes. This element is optional. If this element is omitted, it means that the user does not exist on the node. When this element is set, the SessionID element shall be set for the Core element.

6.2.1.1.3. ResponderUserID element

The ResponderUserID element sets the ID of the user to whom a per-user session is initiated between PUCC nodes. This element is optional. If this element is omitted, it means that the user does not exist on the node. When this element is set, the SessionID element shall be set for the Core element.

6.2.1.2. HelloResponse message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
HelloResponse	Following XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
Result	"Success","Failure"	-	-	1	Required	
Reason	String	-	-	1	Depend on result	

Table 7: Fields of HelloResponse

The HelloResponse message is mapped to the PUCC Core Protocol as follows.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID (String)	-	-	1	Required	
ReplyID	MsgID (String)	-	-	1	Required	
MsgType	"Response"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID(String)	-	-	1	Required	
TraceRoute	-	-	-	-	unused	
Route	-		-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		Protocol	URI	1	Required	

Table 8: Mapping of HelloResponse message to Core Protocol

The following is a sample of HelloResponse message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <MsgType>Response</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>54321.2002-12-20T16:17:32Z@968749ab-f9bb-2647-7459-f66e56f11234</ReplyID>
  <Destination>
    <Target>968749ab-f9bb-2647-7459-f66e56f11234</Target>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <HelloResponse xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <Result>Success</Result>
    </HelloResponse>
  </MsgBody>
</Core>
```

Figure 11. Sample of HelloResponse message

6.2.1.2.1. HelloResponse element

This element is used to designate the response to the hello message. The content of this element describes the "Success" or "Failure" response code and diagnostic description. If a node has no capability to connect to other nodes, the node sends HelloResponse message with "Failure".

6.2.1.2.2. Result element

The Result element designates the "Success" or "Failure" of a PUCC connection or a PUCC session. This element is essential.

6.2.1.2.3. Reason element

The Reason element is used to notify about reason of failure. This element takes following values. If the Result element value is "Failure", Reason element is required. Don't set this element, if it is not so.

Value	Description
NotAuthenticated	Authentication failure
ConnectionCapabilityExceeded	The node has no capability to connect to other nodes.
IncompatibleMode	The node acts according to the incompatible operation mode of behavior.
UserAccessDenied	User access denied due to an authentication failure
SessionAlreadyEstablished	A same session already established between users.

Table 9: Values of Reason element

6.2.1.3. Sequence of Hello method

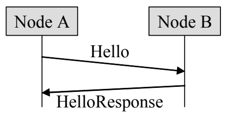


Figure 12. Sequence of Hello method

6.2.2. Bye method

6.2.2.1. Bye message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Bye	-	-	-	1	Required	
		xmlns	URI	1	Required	

Table 10: Fields of Bye message

The Bye message is mapped as per the following table.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Core	Following XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	“Unicast”	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	“Advertise”	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID(String)	-	-	1	Required	
TraceRoute	-	-	-	-	unused	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI		Required	

Table 11: Mapping of Bye message to PUCC Core Protocol

The following is a sample of the Bye message.

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <MsgType>Advertise</MsgType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <Destination>
    <Target>457284ab-f9bb-4305-9900-f98e56f12355</Target>
  </Destination>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <ComType>Unicast</ComType>
  <MsgBody protocol=” Namespace of PUCC Basic Communication Protocol” >
    <Bye xmlns=” Namespace of PUCC Basic Communication Protocol” />
  </MsgBody>
</Core>
```

Figure 13. Sample of Bye message

6.2.2.1.1. Bye element

The Bye element is used to release a PUCC connection from one node to another node or a PUCC session from one user to another user on the PUCC network. This element has no content.

6.2.2.2. Sequence of Bye method

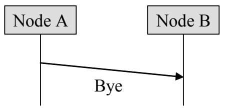


Figure 14. Sequence of Bye method

6.2.3. Resource Information Exchange method

6.2.3.1. ResourceInfomationRequest message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
ResourceInformationRequest	Following XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ResourceQuery	Condition element	-	-	1	Optional	
Condition	-	-	-	1	Optional	
		property	Item of resource	1	Required	
		value	String	1	Required	

Table 12: Fields of ResourceInformationRequest message

The ResourceInformationRequest message is mapped following table.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Core	Following XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast","Broadcast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Request"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route,Target element	-	-	1	ComType Dependent	
Route	Route,Target element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	XML fragment	-	-	1	Required	
HopCount	Hop count(Integer)	-	-	1	Optional	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 13: Mapping of ResourceInformationRequest to PUCC Core Protocol

The following is a sample of the ResourceInformationRequest message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Broadcast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f1235</Source>
  <HopCount>10</HopCount>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <ResourceInformationRequest xmlns=" Namespace of PUCC Basic Communication Protocol" />
  </MsgBody>
</Core>
```

Figure 15. Sample of ResouceInformationRequest

6.2.3.1.1 ResourceInfomationRequest element

The ResourceInformationRequest element is used to request the resource information of a node. This element has no content.

6.2.3.1. ResourceInfomationResponse message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	Description
ResourceInformationResponse	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ResourceData	XML fragment	-	-	1	Required	

Table 14: Fields of ResourceInformationResponse

The ResourceInformationResponse message is mapped following table.

Element name	Element Value	Attribute name (if it has)	Attribute Value	Occurrence	Status	Description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	1	Required	
MsgType	"Response"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route,Target element	-	-	1	Required	
Route	Route,Target element	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	XML fragment	-	-	1	Required	
Route	-	-	-	1	Optional	
		Node	Node ID(String)	1	Required	
HopCount	Hop count (Integer)	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 15: Mapping of ResourceInformationResponse message to PUCC Core Protocol

The following is a sample of the ResourceInformationResponse message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</MsgID>
  <ReplyID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</ReplyID>
  <MsgType>Response</MsgType>
  <Source>nd2</Source>
  <Destination>
    <Target>nd1</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Basic Communication Protocol" >
    <ResourceInformationResponse xmlns=" Namespace of PUCC Basic Communication Protocol" >
      <ResourceData>

      (For details see 6.2.4 Resource Information.)

      </ ResourceData>
    </ResourceInformationResponse>
  </MsgBody>
</Core>
```

Figure 16. Samples of ResourceInformationResponse

6.2.3.1.2. ResourceInformationResponse element

This element designates a response to the ResourceInformationRequest message. The content of this element describes the resource information on the requested node.

6.2.3.1.3. ResourceData element

The ResourceData element is used to describe the resource information of a node. This element has 5 elements. (For details see 6.2.4 Resource Information.)

6.2.3.2. ResourceInformationAdvertise message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
ResourceInformationAdvertise	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ResourceData	XML fragment	-	-	Multiple	Required	

Table 16: Fields of ResourceInformationAdvertise

The ResourceInformationAdvertise message is mapped as shown in the following table.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast", "Broad cast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Advertise"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	

Destination			Route, Target element	-	-	1	ComType Dependent	
	Route		Route , Target Element	-	-	1	Optional	
				Node	Node ID(String)	1	Required	
		Target	Node ID(String)	-	-	1	Required	
TraceRoute			Route element	-	-	1	Required	
	Route		Route element	-	-	1	Optional	
				Node	Node ID(String)	1	Required	
HopCount			Hop Count(Integer)	-	-	1	Optional	
SessionID			Session ID(String)	-	-	1	Optional	
MsgBody			XML fragment	-	-	1	Required	
				protocol	URI	1	Required	

Table 17: Mapping of ResourceInformationAdvertise to PUCC Core Protocol

The following is a sample of ResourceInformationAdvertise message.

```
<Core xmlns= “Namespace of PUCC Core Protocol “>
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687</MsgID>
  <MsgType>Advertise</MsgType>
  <Source>968985ab-e6aa-5842-1234-f98e56f15687</Source>
  <Destination>654742bb-11aa-2586-1123-b00e56f58974</Destination>
  <MsgBody protocol= “Namespace of PUCC Basic Communication Protocol “>
    <ResourceInformationAdvertise xmlns= “Namespace of PUCC Basic Communication Protocol “>
      <ResourceData>

      (For details see 6.2.4 Resource Information.)

      </ ResourceData>
    </ResourceInformationAdvertise>
  </MsgBody>
</Core>
```

Figure 17. Sample of ResourceInformationAdvertise

6.2.3.3.1. ResourceInformationAdvertise element

This element designates an advertise message sent to other nodes. The contents of this element describe the resource information on the node.

6.2.3.3.2. ResourceData element

The ResourceData element is used to describe resource information of a node. This element has 5 elements. (For details see 6.2.4 Resource Information.).

6.2.3.4. Sequence of Resource Information Exchange method

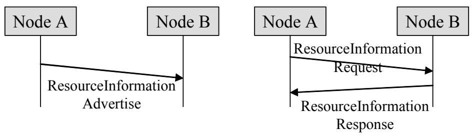

Node A
Node B
ResourceInformation Advertise
Node A
Node B
ResourceInformation Request
ResourceInformation Response

Figure 18. Sequence of Resource Information Exchange method

6.2.4. Resource Information

6.2.4.3. Items of Resource Information

Information of each node in a PUCC network (Network Metadata) is described using XML. The node requests another node’s information, and the response is made using XML. The above process flow is named “Resource Information Exchange”. The following are definitions of the fields in Resource Information.

Items	Description
OwnNodeID	Own Node ID
OwnedApplication	The Functions (application) provided by the node (e.g.: search, instant message)
ConnectionCapability	The number of the maximum connection.
ConnectedNode	The Node ID of the connecting nodes. (e.g.: node B, node C)
TransportAddress@protocol	A transport layer protocol of a node.
TransportAddress@type	A transport layer address of a node.

Table 18: Items of Resource Information

6.2.3.1. Schema Definition

The following are definitions of the fields (Parameter) in Resource Information.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
ResourceData	XML Fragment	-	-	Multiple	Required	
OwnNodeID	Node ID	-	-	1	Required	
OwnedApplication	Application element	-	-	1	Optional	
Application	URI (String)	-	-	Multiple	Required	
ConnectionCapability	Maximum number of connection	-	-	1	Required	
ConnectedNode	XML fragment	-	-	1	Required	
Node	TransportAddress element, ConnectionCapability element, RTT element	-	-	Multiple	Optional	
		ID	Node ID(String)	1	Required	
		parent	“true”, “false”	1	Required	
Transport Address	Transport layer address (String)	-	-	Multiple	Required	
		protocol	classification of protocol(String) such as ”TCP”, ”SOAP”, ”HTTP”, ”BEEP”, ”Bluetooth” , “OBEX”, “IEEE1394”	1	Required	
		type	classification of address(String) such as ”IPv4”, ”IPv6”, ”URL”,”Bluetooth”, “IEEE1394”	1	Required	
ConnectionCapability	Maximum number of connection (Integer)	-	-	1	Optional	
RTT	RTT by Ping. Time scale is millisecond. (Integer)	-	-	1	Optional	
TransportAddress	Transport layer	-	-	Multiple	Required	

		address (String)	protocol	classification of protocol(String) such as "TCP", "SOAP", "HTTP", "BEEP", "Bluetooth", "OBEX", "IEEE1394"	1	Required	
			type	classification of address(String) such as "IPv4", "IPv6", "URL","Bluetooth", "IEEE1394"	1	Required	

Table 19: Structure of ResourceData Element

6.2.3.1.1. ResourceData element

The ResourceData element is used to describe resource information of a node. This element has 5 elements.

6.2.3.1.2. OwnNodeID element

The OwnNodeID element is used to designate a node ID.

6.2.3.1.3. OwnedApplication element

The OwnedApplication element is used to describe the name of applications running in a node. This element has multiple Application elements according to the applications running in the node. The content of the Application element is an application identifier described using URI. This element is optional.

Element name		Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
OwnedApplication		XML fragment	-	-	1	Optional	
	Application	URI (String)	-	-	Multiple	Required	

Table 20: The OwnedApplication element

6.2.3.1.4. ConnectionCapability element

The ConnectionCapability element is used to designate the maximum number of connections. A node can specify a number of connections using this element. This element is essential.

6.2.3.1.5. ConnectedNode element

The ConnectedNode element is used to describe a list of nodes connected to a node. This element is essential. This element has Node elements which have TransportAddress elements ConnectionCapability elements, RTT elements and ID attribute.

The ConnectionCapability element is used to designate the maximum number of connections of the comnected nodes.

The RTT element is used to designate a round trip time on connections with the connected nodes.

The TransportAddress element has two attributes: protocol and type. The protocol attribute is used to designate the class of the transport protocol used such as TCP, and SOAP. The type element is used to designate the class of transport address used such as IPv4 and URL.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
ConnectedNode	XML fragment	-	-	1	Required	
Node	TransportAddrtess element, ConnectionCapability element, RTT element	-	-	Multiple	Optional	
		ID	Node ID (String)	1	Required	
		parent	"true","false"	1	Required	Default value is "false".
Transport Address	Transport layer address (String)	-	-	Multiple	Required	
		protocol	classification of protocol(String) such as "TCP", "SOAP", "HTTP", "BEEP", "Bluetooth" , "OBEX", "IEEE1394"	1	Required	
		type	classification of address(String) such as "IPv4", "IPv6", "URL","Bluetooth", "IEEE1394"	1	Required	
Connection Capability	Maximum number of connection (Integer)	-	-	1	Optional	
RTT	RTT by Ping. Time scale is millisecond. (Integer)	-	-	1	Optional	

Table 21: The ConnectedNode element

6.2.3.1.6. TransportAddress element

The TransportAddress element is used to designate the transport layer addresses of the node. This element has two attributes: protocol and type. The protocol attribute is used to designate the class of the transport protocol used such as TCP, and SOAP. The type element is used to designate the class of transport address used such as IPv4 and URL. The contents of the TransportAddress element are the transport addresses of the node. This element is essential.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
TransportAddress	Transport layer address (String)	-	-	Multiple	Required	
		protocol	classification of protocol(String) such as "TCP", "SOAP", "HTTP", "BEEP", "Bluetooth" , "OBEX", "IEEE1394"	1	Required	
		type	classification of address(String) such as "IPv4", "IPv6", "URL","Bluetooth", "IEEE1394"	1	Required	

Table 22: The TransportAddress element

```
<ResourceData>
  <OwnNodeID>968742ab-f9bb-4305-9900-f98e56f12352</OwnNodeID>
  <OwnedApplication>
    <Application>http://www.xxx.co.jp/PUCC_search_app</Application>
    <Application>http://www.xxx.co.jp/PUCC_instantmsg_app</Application>
  </OwnedApplication>
  <ConnectionCapability>5</ConnectionCapability>
  <ConnectedNode>
    <Node ID=" 145742a8-63cc-7569-5468-f98e55f12478" >
      <TransportAddress protocol=" TCP"  type=" IPv4" >10.74.126.155<TransportAddress>
    </Node>
    <Node ID= "654742bb-11aa-2586-1123-b00e56f58974 ">
      <TransportAddress protocol=" TCP"  type=" IPv4" >10.74.126.156<TransportAddress>
    </Node>
  </ConnectedNode>
  <TransportAddress protocol=" TCP"  type=IPv4" >192.168.0.56</TransportAddress>
</ ResourceData>
```

Figure 19. A sample of ResourceData

The DTD of node resource information is described in APPENDIX E.

6.5. PUCC Control Message Protocol

The PUCC Control Message Protocol provides ancillary functions such as those to notify message forwarding errors, to keep a PUCC connection or session alive, diagnose the condition of a node and discover the first PUCC node.. The PUCC Control Message Protocols include;

- Error Report method

The Error Report method is used to notify a source node of the forwarding error of a message The Error Report method is a Proactive (Advertise) type method and involves an ErrorReport message.

- Diagnose method

The Diagnose method is used to measure RTT (Round Trip Time) between PUCC nodes, keep a PUCC connection or session alive and search a route to the PUCC node satisfied the condition of a node. The Diagnose method is a Reactive (Request/Response) type method and involves a Diagnose message and a DiagnoseResponse message.

- Lookfor method

The Lookfor method is used to find the first PUCC node to which a PUCC node can connect in the light PUCC network. When a PUCC node receives a Lookfor message, if the community ID of the message matches the community ID of a community that it is participating in, the PUCC node responds with a LookforResponse message. The Lookfor method is a Reactive (Request/Response) type method and involves a Lookfor message and a LookforResponse message.

6.5.1. Error Report method

6.5.1.1. ErrorReport message

The ErrorReport message has the following structure.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
ErrorReport	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ErrorMsgID	Message ID of an error message (String)	-	-	1	Required	
Error	String	-	-	1	Required	

Table 23: Fields of ErrorReport message

6.5.1.1.1. ErrorReport element

The ErrorReport element is used to designate an error in message forwarding. This element has an ErrorMsgID element and an Error element. This element is essential.

6.5.1.1.2. ErrorMsgID element

The ErrorMsgID element is used to designate the Message ID of an error message. This element has a string representing the Message ID. This element is essential.

6.5.1.1.3. Error element

The Error element is used to provide a description of an error. The following table shows error descriptions. This element is essential.

No.	Error	Description
1	StaleRoutingInformation	This error indicates that message destination is wrong. (The message forwarding is success)
2	DestinationUnreachable	This error designates failure of message forwarding.
3	ParameterProblem	Format of PUCC Core Protocol is wrong.
4	SourceQuench	Message abandoned due to congestion.
5	NodeFailure	This error designates failure of message forwarding to adjacent nodes.
6	UnsupportedProtocol	Correspondent protocol is unsupported.

Table 24: Error items

The following shows the mapping of ErrorReport message to PUCC Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Advertise"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	XML fragment	-	-	1	Required	
Route	Route, Target element	-	-	Multiple	Optional	
		Node	Node ID (Sting)	1	Required	
Target	Node ID (String)	-	-	Multiple	Required	
TraceRoute	XML fragment	-	-	1	Optional	
Route	Route element	-	-	Multiple	Optional	
		Node	Node ID (Sting)	1	Required	
HopCount	Hop count (Integer)	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 25: Mapping of ErrorReport message to Core Protocol

The following shows a sample of the ErrorReport message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <MsgType>Advertise</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>968985ab-e6aa-5842-1234-f98e56f15687</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <ErrorReport xmlns=" Namespace of PUCC Control Message Protocol" >
      <ErrorMsgID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</ErrorMsgID>
      <Error>DestinationUnreachable</Error>
    </ErrorReport>
  </MsgBody>
```

```
</Core>
```

Figure 20. A sample of ErrorReport message

6.5.1.2. Sequence of ErrorReport method

The ErrorReport method is a proactive communication mode and involves just an advertise type message.

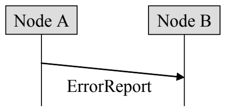


Figure 21. Sequence of ErrorReport service

6.5.2. Diagnose method

6.5.2.1. Diagnose message

The Diagnose message has the following structure.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Diagnose	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
DiagnoseData	Sting	-	-	1	Optional	
DiagnoseDestination	ID (String)	-	-	1	Optional	
		type	NodeID, ApplicationID	1	Required	

Table 26: Fields of Diagnose message

6.5.2.1.1. Diagnose element

The Diagnose element is used to request a response from another PUCC node. This element has DiagnoseData and DiagnoseDestination element. This element is essential.

6.5.2.1.2. DiagnoseData element

The DiagnoseData element is used to set an arbitrary message to diagnose. If RTT (Round Trip Time) between PUCC nodes is measured, this element has the number of milliseconds from Jan. 1st, 1970. This element is optional.

6.5.2.1.3. DiagnoseDestination element

The DiagnoseDestination element is used to designate a search condition of a node. This element has a type attribute. The following table shows condition descriptions. This element is optional.

No.	Condition	Description
1	NodeID	Search a node which has correspondent node ID
2	ApplicationID	Search a node which has correspondent application

Table 27: Condition items

The following shows the mapping of Diagnose message to PUCC Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description

Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast","Broadcast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	-	unused	
MsgType	"Request"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route, Target element	-	-	1	ComType Dependent	
Route	Route, Target element	-	-	1	Optional	
		Node	Node ID (Sting)	1	Required	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	Route element	-	-	1	Required	
Route	Route element	-	-	Multiple	Optional	
		Node	Node ID (Sting)	1	Required	
HopCount	Hop count (Integer)	-	-	1	Optional	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 28: Mapping of Diagnose message to Core Protocol

The following shows a sample Diagnose message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Broadcast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <Diagnose xmlns=" Namespace of PUCC Control Mesage Protocol" >
      <DiagnoseData>104353920000</DiagnoseData>
      <DiagnoseDestination type=" NodeID" >968985ab-e6aa-5842-1234-f98e56f15687</DiagnoseDestination>
    </Diagnose>
  </MsgBody>
</Core>
```

Figure 22. A sample of Diagnose message

6.5.2.2. DiagnoseResponse message

The DiagnoseResponse message has the following structure.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
DiagnoseResponse	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
DiagnoseData	Sting	-	-	1	Optional	

Table 29: Fields of Diagnose message

6.5.2.2.1. DiagnoseResponse element

The DiagnoseResponse element is used to respond to a Diagnose message. This element has a DiagnoseData element. This element is essential.

6.5.2.2.2. DiagnoseData element

The DiagnoseData element is used to designate echo of DiagnoseData element of Diagnose message. This element is optional. If DiagnoseData element in the Diagnose message is specified, this element is required.

The following shows the mapping of DiagnoseResponse message to PUCC Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	“Unicast”	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	1	Required	
MsgType	“Response”	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route, Target element	-	-	1	Required	
Route	Route, Target element	-	-	Multiple	Optional	
		Node	Node ID (Sting)	1	Required	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	XML fragment	-	-	1	Required	
Route	Route, Target element	-	-	Multiple	Optional	
		Node	Node ID (Sting)	1	Required	
HopCount	Hop count (Integer)	-	-	-	unused	
SessionID	Session ID(String)	-	-	1	Optional	
MsgBody	XML fragment	-	-	1	Required	
		protocol	URI	1	Required	

Table 30: Mapping of DiagnoseResponse message to Core Protocol

The following shows a sample DiagnoseResponse message.

```
<Core xmlns=” Namespace of PUCC Core Protocol” >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <ReplyID>12345.2002-12-20T16:15:32Z@968742ab-f9bb-4305-9900-f98e56f12352</ReplyID>
  <MsgType>Response</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>968985ab-e6aa-5842-1234-f98e56f15687</Target>
  </Destination>
  <MsgBody protocol=” Namespace of PUCC Control Message Protocol” >
    <DiagnoseResponse xmlns=” Namespace of PUCC Control Message Protocol” >
      <DiagnoseData>104353920000</DiagnoseData>
    </DiagnoseResponse>
  </MsgBody>
</Core>
```

Figure 23. A sample of DiagnoseResponse message

6.5.2.3. Sequence of Diagnose method

The Diagnose method is a reactive communication mode and involves a request message and a response message.

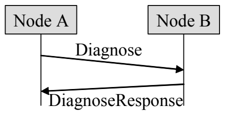


Figure 24. Sequence of Diagnose method

6.5.3. Lookfor method

6.5.3.1. Lookfor message

The Lookfor message has the following structure.

Element name		Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Lookfor		XML fragment	-	-	1	Required	
			xmlns	URI	1	Required	
			count	The number of LookforResponse message.(Integer)	1	Optional	
	Condition	Original search condition of application.(String)	-	-	1	Optional	
			protocol	URI	1	Required	

Table 31: Fields of Lookfor message

6.5.3.1.1. Lookfor element

The Lookfor element is used to locate PUCC nodes in a network. This element has a count attribute and a Condition element. The count attribute is optional and used to designate the request of the number of LookforResponse messages from the PUCC network. If the count attribute is omitted, the value is regarded as infinite. This element is essential.

6.5.3.1.2. Condition element

When a node receives a Llookfor message with a Condition element, the node notifies application of the namespace of the Condition element specified. This element is used to search a node which satisfies the conditions that the application specifies.

The following shows the mapping of a Lookfor message to the PUCC Core Protocol. This element is optional.

Element name			Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Core			XML fragment	-	-	1	Required	
				xmlns	URI	1	Required	
	ComType		“Broadcast”	-	-	1	Required	
	MsgID		Message ID(String)	-	-	1	Required	
	ReplyID		Message ID(String)	-	-	-	unused	
	MsgType		“Request”	-	-	1	Required	
	CommunityID		Community ID(String)	-	-	1	Optional	
	Source		Node ID(String)	-	-	1	Required	
	Destination		-	-	-	-	unused	
		Route	-	-	-	-	unused	

		Node	Node ID(String)	-	unused	
Target	-	-	-	-	unused	
TraceRoute	-	-	-	-	unused	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
SessionID	Session ID(String)	-	-	-	unused	
MsgBody	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	

Table 32: Mapping of Lookfor message to Core Protocol

The following shows a sample Lookfor message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Broadcast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <Lookfor xmlns=" Namespace of PUCC Control Message Protocol" />
  </MsgBody>
</Core>
```

Figure 25. A sample of Lookfor message

6.5.3.2. LookforResponse message

The LookforResponse message has the following structure.

Table 33: Fields of LookforResponse message

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
LookforResponse	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
Description	String	-	-	1	Optional	
DeviceList	XML fragment	-	-	1	Optional	
Device	XML fragment	-	-	multiple	Optional	
		type	Device ID(URI)	1	Required	
		id	Globally unique ID of the device (String)	1	Optional	
		name	Short user-friendly name of the device (String)	1	Optional	

6.5.3.2.1. LookforResponse element

The LookforResponse element is used to respond to a Lookfor message. This element is essential.

6.5.3.2.2. Description element

The Description element configures an arbitrary message that the node or application which has responded to a search wants to report to the source of the search. This element is optional.

6.5.3.2.3. DeviceList element

The DeviceList element is used by the node or application which has responded to a search to configure a list of the devices that exist in the node.This element is optional.

6.5.3.2.4. Device element

The Device elements sets static metadata for a single device. This element is Optional. This element has type, id and name attributes. The type attribute designates the device type. The type attribute is essential. The id attribute designates the globally unique id for the device and should be < 32 characters. The id attribute is optional. The name attribute designates the short user-friendly name of the device and should be < 64 characters. The name attribute is optional.

The Device element has the following structure.

Table 34: Fields of Device element

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Device	XML fragment	-	-	1	Optional	
		type	Device ID (URI)	1	Required	
		id	Globally unique ID of the device （String）	1	Optional	
		name	Short user-frindly name of the device (String)	1	Optional	
URLBase	URL (RFC2396)	-	-	1	Optional	
Manufacturer	String	-	-	1	Optional	
ManufacturerURL	URL (RFC2396)	-	-	1	Optional	
ManufactureDate	Date (ISO8601)	-	-	1	Optional	
ModelDescription	String		-	1	Optional	
ModelName	String	-	-	1	Optional	
ModelNumber	String	-	-	1	Optional	
ModelURL	URL (RFC2396)	-	-	1	Optional	
SerialNumber	String	-	-	1	Optional	
UDN	Unique Device Name(URI)	-	-	1	Optional	
UPC	String.	-	-	1	Optional	
IconList	XML fragment	-	-	1	Optional	
Icon	XML fragment	-	-	multiple	Optional	

A) URLBase element

The URLBase element designates the Base URL in the Device element content. It is used to construct fully-qualified URLs. All relative URLs that appear elsewhere in the Device element are combined with this base URL according to the rules in RFC2396. This element is optional.

B) Manufacturer element

The Manufacturer element designates the manufacturer name of the device. It is string and should be < 64 characters. This element is optional.

C) ManufacturerURL element

The ManufacturerURL designates the URL to manufacture site of the device. It is specified by each vendor. This element is optional.

D) ManufactureDate element

The ManufactureDate designates the date of manufacture of the device. It follows ISO8601 date format. This element is optional.

E) ModelDescription element

The ModelDescription element designates the long user-friendly name of the device. It is specified by each vendor. It is string and should be < 128 characters. This element is optional.

F) ModelName element

The ModelName element designabtes the Model name of the device. It is specified by each vendor. It is string and should be < 32 characters. This element is optional.

G) ModelNumber element

The ModelNumber element designates the model number of the device. It is specified by each vendor. It is string and should be < 32 characters. This element is optional.

H) ModelURL element

The ModelURL element designates the URL to model site of the device. It is specified by each vendor. This element is optional.

I) SerialNumber element

SerialNumber element designates the manufacture's serial number of the device. It is specified by each vendor. This element is optional.

J) UDN element

The UDN element designates the Unique Device Name of the device. It is a universally-unique identifier for the device. It must begin with uuid:. It is specified by each vendor. This element is optional.

K) UPC element

The UPC element designates the Universal Product Code of the device. It is a 12-digit string and all-numeric code that identifies the consumer package. It is specified by each vendor. This element is optional.

L) IconList element

The IconList element configures a list of icon data that the device has. This element is optional.

M) Icon element

Icon element configures one a set of icon data. This element is optional.

The Icon element has the following structure.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Icon	XML fragment	-	-	1	Optional	
Mimetype	String (RFC2045, 2046, 2387)	-	-	1	Optional	
Width	Integer	-	-	1	Optional	
Height	Integer	-	-	1	Optional	
Depth	Integer	-	-	1	Optional	
Url	URL (RFC2396)	-	-	1	Optional	

Table 35: Fields of Icon element

N) Mimetype element

The Mimetype element designates the icon's MIME typ**e.** This element is optional.

O) Width element

THe Width element designates the horizontal dimension of the icon in pixels. It is an integer. This element is optional.

P) Height element

The Height element designates the vertical dimension of the icon in pixels. It is an integer. This element is optional.

Q) Depth element

The Depth element designates the number of color bits per pixel of the icon. It is an integer. This element is optional.

R) Url element

The Url element designates the ID for the Icon Image (URL). This element is optional.

The following shows the mapping of the LookforResponse message to PUCC Core Protocol.

Element name	Element Value	Attribute name (if present)	Attribute Value	Occurrence	Status	description
Core	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	
ComType	"Unicast"	-	-	1	Required	
MsgID	Message ID(String)	-	-	1	Required	
ReplyID	Message ID(String)	-	-	1	Required	
MsgType	"Response"	-	-	1	Required	
CommunityID	Community ID(String)	-	-	1	Optional	
Source	Node ID(String)	-	-	1	Required	
Destination	Route, Target element	-	-	1	Required	
Route	-	-	-	-	unused	
		Node	Node ID(String)	-	unused	
Target	Node ID (String)	-	-	1	Required	
TraceRoute	-	-	-	-	unused	

Route	-		-	-	unused	
		Node	Node ID(String)	-	unused	
HopCount	Hop count (Integer)	-	-	-	unused	
SessionID	Session ID(String)	-	-	-	unused	
MsgBody	XML fragment	-	-	1	Required	
		xmlns	URI	1	Required	

Table 36: Mapping of LookforResponse message to Core Protocol

The following show a sample LookforResponse message.

```
<Core xmlns=" Namespace of PUCC Core Protocol" >
  <ComType>Unicast</ComType>
  <MsgID>12345.2002-12-20T16:17:00Z@968985ab-e6aa-5842-1234-f98e56f15687 </MsgID>
  <MsgType>Request</MsgType>
  <Source>968742ab-f9bb-4305-9900-f98e56f12352</Source>
  <Destination>
    <Target>968985ab-e6aa-5842-1234-f98e56f15687</Target>
  </Destination>
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <LookforResponse xmlns=" Namespace of PUCC Control Message Protocol" />
  </MsgBody>
</Core>
```

Figure 26. A sample of LookforResponse message

6.5.3.3. Sequence of Lookfor method

The Lookfor method is a reactive communication mode and involves a request message and a response message.

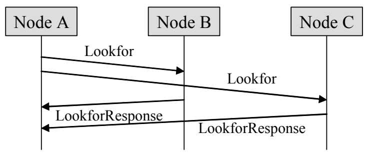


Figure 27. Sequence of Lookfor method

7. Transport protocol bindings

This section describes the transport bindings. We defined all PUCC protocols independent of the underlying layer. In this specification, TCP/UDP binding, Bluetooth binding, IEEE1394 binding, OBEX binding, and HTTP binding are defined.

7.1. TCP Transport

7.1.1. Frame

This section describes the transport binding of the PUCC Protocols over TCP/IP. All PUCC messages are encapsulated by the following frame when the PUCC Protocol is used over TCP/IP.

	Description	Status

Header	There are 2 kinds of header.
	A) Frame of normal message
	`P2PFRM<SP>Connectiontype<SP>FrameNo(- SeqNo/WholePacketCounts) <SP>Size<SP>DestNodeID<SP>SrcNodeID ( <SP>CommunityID ) <CR><LF>`
	B) Frame in response to a frame error
	`FRMERR<CR><LF>`
		Required

	`Connectiontype` The Connectiontype parameter is used to designate a connection type. `0: keep-alive` (TCP connection not disconnected after this frame.) `1: single` (Disconnect TCP connection after this frame.)	Required
	`FrameNo` The FrameNo parameter is sequential number used to distinguish frames. This parameter is set at the default value at 0 and final value is 2147483647. If a PUCC message is divided, "-SeqNo/WholePacketCounts" is given to the end of FrameNo. The frame is specified by changing only the part of "- SeqNo/WholePacketCounts". The position of a frame is specified by "-SeqNo/WholePacketCounts".	Required
	`Size` The Size parameter is used to designate size of payload (without header and trailer) by the octet number.	Required
	`DestNodeID` The DestNodeID parameter is used to designate destination node ID.	Required
	`SrcNodeID` The SrcNodeID parameter is used to designate source node ID.	Required
	`CommunityID` The CommunityID parameter is used to distinguish which community the message belongs to.	Optional

Payload	A) Frame of normal message
	`(MIME entity header) <CR><LF> [PUCC message]`
	The MIME entity header is optional. The Payload has one PUCC message.
	B) Frame to response to a frame error
	`[Header of P2PFRM]`
	The Payload has one header of P2PFRM which is regarded as error.
		Required

Trailer	`FRMEND<CR><LF>`	Required

Table 37: The format of the frame for binding of PUCC Protocols over TCP/IP

The following shows a sample P2PFRM frame.

```
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:DoCoMo

<Core xmlns=" Namespace of PUCC Core Protocol" >
 //omitted
</Core>
FRMEND
```

Figure 28. A sample of P2PFRM frame over TCP/IP

The following shows a sample FRMERR frame.

```
FRMERR
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:DoCoMo
FRMEND
```

Figure 29. A sample of FRMERR frame over TCP/IP

7.1.2. Connection Type

We defined two connection types for the transport binding of the PUCC Protocols over TCP/IP.

1. **Persistent connection**

   PUCC nodes which connect to each other keep the TCP connection active and send PUCC messages using it while the PUCC connection is active. When PUCC connection is released, TCP connection is released at once.

2. **Separate connection**

   PUCC node sends a PUCC message using one TCP connection and closes it after sending the message.

When PUCC nodes establish a PUCC connection, they decide the connection type. The PUCC nodes negotiate the connection type, using the connectiontype parameter of a frame. The PUCC node sends a Hello message with the value of connectiontype, the other PUCC node sends a HelloResponse message with the value of connectiontype. Requesting a keep-alive connection is regarded as a higher level request than a single connection.

For example, if a PUCC node receives a Hello message with a request for a keep-alive connection, the PUCC node can select either connection type, keep-alive or single. Conversely, if a PUCC node receives a Hello message with a request for a single connection, the PUCC node can chose only a single connection. The following table shows cases of negotiation on connection type.

Case	Value of connectiontype in the Hello message.	Value of connectiontype in the HelloResponse message.	Determined connection type
1	0 : keep-alive	0 : keep-alive	0 : keep-alive
2	0 : keep-alive	1 : single	1 : single
3	1 : single	1 : single	1 : single

Table 38: Cases of negotiation on connection type

According to the above negotiation, the TCP binding handler works as follows.

Case	Value of connectiontype in the Hello message.	Value of connectiontype in the HelloResponse message.	Destination of a HelloResponse message

1	0 : keep-alive	0 : keep-alive	Source address and port of Hello message (Use same socket)
		1 : single	Source address of Hello message and well-known port. (Open another socket)
2	1 : single	1 : single	Source address of Hello message and well-known port. (Open another socket)

Table 39: Selection of port number

The TCP binding handler closes a TCP connection upon receiving a Bye message.

If PUCC connection establishment is unsuccessful, the PUCC node returns a HelloResponse message with a request for a single connection.

7.2. UDP/IP Transport

7.2.1. Frame

This section describes the transport binding of the PUCC Protocols over UDP/IP. The Lookfor message and LookforResponse of PUCC Control Message Protocol are encapsulated with the following frame when the PUCC Protocol is used over UDP/IP. The message frame is almost the same as that of TCP/IP. Differences are the Connectiontype parameter and the DestNodeID parameter. The Connectiontype is only "1" because UDP is used. The DestNodeID is specified as "*" when Lookfor message is sent using IP Multicast.

	Description	Status

Header	There are 2 kinds of header.
	C) Frame of normal message
	`P2PFRM<SP>Connectiontype<SP>FrameNo(-SeqNo/WholePacketCounts)<SP>Size`
	`<SP>DestNodeID<SP>SrcNodeID ( <SP>CommunityID ) <CR><LF>`
	D) Frame to response to a frame error
	`FRMERR<CR><LF>`
		Required

	`Connectiontype` `1: single` (Using UDP)	Required
	`FrameNo` The FrameNo parameter is a sequential number used to distinguish frames. This parameter is set at the default value at 0 and the final value is 2147483647. If a PUCC message is divided, "-SeqNo/WholePacketCounts" is given to the end of FrameNo. The frame is specified by changing only the part of "- SeqNo/WholePacketCounts". The position of a frame is specified by "-SeqNo/WholePacketCounts".	Required
	`Size` The Size parameter is used to designate size of payload (without header and trailer) by the octet number.	Required
	`DestNodeID` The DestNodeID parameter is used to designate destination node ID. If IP Multicast is used, "*" is specified.	Required
	`SrcNodeID` The SrcNodeID parameter is used to designate source node ID.	Required
	`CommunityID` The CommunityID parameter is used to distinguish which community the message belongs to.	Optional

Payload	C) Frame of normal message
	`(MIME entity header) <CR><LF> [PUCC message]`
	The MIME entity header is optional. The Payload has one PUCC message.
	D) Frame to response to a frame error
	`[Header of P2PFRM]`
	The Payload has one header of P2PFRM which is regarded as error.
		Required

Trailer	`FRMEND<CR><LF>`	Required

Table 40: The format of the frame for binding of PUCC Protocols over UDP/IP

The following shows a sample of the P2PFRM frame over UDP/IP.

```
P2PFRM 0 0-1/3 702 * 113541a8-bb56-8746-b456c457581 urn:pucc:community:DoCoMo

<Core xmlns=" Namespace of PUCC Core Protocol" >
 //omitted
  <MsgBody protocol=" Namespace of PUCC Control Message Protocol" >
    <Lookfor xmlns=" Namespace of PUCC Control Message Protocol" />
  </MsgBody>
</Core>
FRMEND
```

Figure 30. A sample of P2PFRM frame over UDP/IP

7.3. Bluetooth Transport

7.3.1. L2CAP

Bluetooth transport uses the L2CAP connection channel to send and receive PUCC messages. L2CAP packet has a 3 part structure and payload of L2CAP packet is bound to PUCC Protocols.

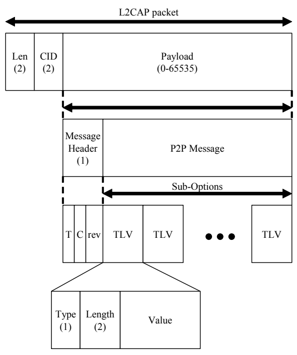


Figure 31. Transport binding of PUCC protocols over Bluetooth

The following shows the detailed structure of PUCC message over L2CAP.

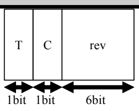
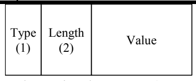
	Description	Status
Header		Required
	`Frame Type(T) : 1bit` This bit must be set to 1.	Required
	`Connection Type(C) : 1 bit` This bit indicates whether Bluetooth connection (relation of Master-Slave) must be kept or not, If C is 0, the connection might be kept if possible. If C is 1, the connection must be closed. The detailed behavior of this bit is identical to that described in Section 7.1.1.2.	Required
	`Reserved (RSV) : 6bit` Reserved bits	Required
Sub-Option	 Sub-Option has TLV (Type-Length-Value) structure. Type and Length fields are statically 1 byte and 2 bytes long, respectively. The Value field length is specified in Length field in units of bytes. There are currently three types defined.	Required
	`0x01 (TYPE_XML_DATA)` A PUCC message ranging <Core> tag to </Core> tag in bytes of UTF-8	Required
	`0x02 (TYPE_SRC_NODEID)` A source node's Node ID in bytes of UTF-8.	Required
	`0x03 (TYPE_DST_NODEID)` A Destination node's Node ID in bytes of UTF-8	Optional

Table 41: Message Format for Bluetooth L2CAP Transport

7.3.2. SPP

Bluetooth transport uses the SPP connection channel to send and receive PUCC messages.

7.3.2.1. Frame

7A3260162A 2009/7/2 12:38
書式変更: 箇条書きと段落番号

The following shows the detailed structure of PUCC message over SPP.

	Description	Status

Header	There are 2 kinds of header.
	A) Frame of normal message
	`P2PFRM<SP>Connectiontype<SP>FrameNo(- SeqNo/WholePacketCounts) <SP>Size<SP>DestNodeID<SP>SrcNodeID ( <SP>CommunityID ) <CR><LF>`
	B) Frame in response to a frame error
	`FRMERR<CR><LF>`
		Required

	`Connectiontype` The Connectiontype parameter is used to designate a connection type. `0: keep-alive` (connection not disconnected after this frame.)	Required
	`FrameNo` The FrameNo parameter is sequential number used to distinguish frames. This parameter is set at the default value at 0 and final value is 2147483647. If a PUCC message is divided, "-SeqNo/WholePacketCounts" is given to the end of FrameNo. The frame is specified by changing only the part of "- SeqNo/WholePacketCounts". The position of a frame is specified by "-SeqNo/WholePacketCounts".	Required
	`Size` The Size parameter is used to designate size of payload (without header and trailer) by the octet number.	Required
	`DestNodeID` The DestNodeID parameter is used to designate destination node ID. If destination node ID is not fixed, "*" is specified.	Required
	`SrcNodeID` The SrcNodeID parameter is used to designate source node ID.	Required
	`CommunityID` The CommunityID parameter is used to distinguish which community the message belongs to.	Optional

Payload	A) Frame of normal message
	`(MIME entity header) <CR><LF> [PUCC message]`
	The MIME entity header is optional. The Payload has one PUCC message.
	B) Frame to response to a frame error
	`[Header of P2PFRM]`
	The Payload has one header of P2PFRM which is regarded as error.
		Required

Trailer	`FRMEND<CR><LF>`	Required

Table 42: Message Format for Bluetooth SPP Transport

7A3260162A 2009/7/2 12:38
書式変更: 箇条書きと段落番号

7A3260162A 2009/7/2 12:38
書式変更: 箇条書きと段落番号

7A3260162A 2009/7/2 12:38
書式変更: 箇条書きと段落番号

7A3260162A 2009/7/2 12:38
書式変更: 箇条書きと段落番号

7.3.2.2. Connection Type

7A3260162A 2009/7/2 12:39
書式変更: 箇条書きと段落番号

When PUCC protocol is used over Bluetooth SPP Transport, ConnectionType value is 0 (keep-alive) .

7.3.2.3. PUCC セッションと仮想シリアル接続の関係

7A3260162A 2009/7/2 14:44
書式変更: 箇条書きと段落番号

PUCC セッションを開設する側が Bluetooth SPP の Device A の役割（仮想シリアル接続を行う）を担う。PUCC セッションを開設される側が Bluetooth SPP の Device B の役割（仮想シリアル接続を待ち受ける）を担う。PUCC セッションは、１つの仮想シリ

アル接続（1 つの L2CAP チャネル上の 1 つの RFCOMM セッション）上に開設する。

7.3.2.4. 仮想シリアル通信パラメータ

仮想シリアル通信のための以下のパラメータの設定値は本仕様書では規定しない。あらかじめノード間で合意されているという前提とする。

- 通信速度（ボーレート）
- データビット数
- ストップビット数
- パリティ（無し／奇数／偶数／マーク／スペース）
- フローコントロール（有／無、入力／出力）

7.3.2.5. ペアリング

本仕様には、ペアリングの方法は規定しない。あらかじめノード間で通信相手の BD アドレス（Bluetooth Device Address）とパスキーを知っていることを前提とする。

7A3260162A 2009/7/2 14:44
書式変更: 箇条書きと段落番号

7A3260162A 2009/7/2 14:44
書式変更: 箇条書きと段落番号

7A3260162A 2009/7/2 14:44
書式変更: 箇条書きと段落番号

7.4. IEEE1394 Transport

This section describes the transport binding of the PUCC Protocols over IEEE1394. All PUCC messages are encapsulated by the following frame when the PUCC Protocol is used over IEEE1394.

7.4.1. Frame

The message frame is the same as TCP/IP.

7.4.2. Communication Method

There are two communication methods in IEEE1394. First is an Asynchronous communication. In the Asynchronous communication, it is guaranteed to transmit the packet to the other node surely. However, it isn't guaranteed to delay a transmission. Second is an Isochronous communication. This communication suits a transmission of video or voice data. In the Isochronous communication, it is guaranteed to complete the data transmitting every 125[μ sec]. PUCC Protocols over IEEE1394 basically treats only a control message. A binary data like multimedia contents etc. is not treated. Therefore, the message is sent and received by using IEEE1394 Asynchronous Transmission.

7.4.3. Configuration ROM

A specification of Configuration Rom uses a specification that defined IP over 1394.

7.4.4. Address Resolution

In the address of IEEE1394, There is two ID, IEEE1394 node ID (16bit variable) and EUI-64 address (64bit fixed). IEEE1394 node ID is changed by bus reset. However, EUI-64 address isn’t changed by bus reset, because of equipment unique. Therefore, the transport address of PUCC node uses EUI-64 address. In the IEEE1394 network, IEEE1394 node ID is used. So, each node has a corresponding table of EUI-64 address and IEEE1394 node ID. And, each node renews this corresponding table at each bus reset.

7.5. OBEX Transport

7.5.1. Frame

This section describes the transport binding of the PUCC Protocols over OBEX. All PUCC messages are encapsulated by the following frame when the PUCC Protocol is used over OBEX.

Table 1: The format of the frame for binding of PUCC Protocols over OBEX

	Description		Status

Header	There are 2 kinds of header.
	E) Frame of normal message
	`P2PFRM<SP>Connectiontype<SP>FrameNo(- SeqNo/WholePacketCounts)<SP>Size<SP>DestNodeID<SP>SrcNodeID ( <SP>CommunityID ) <CR><LF>`
	F) Frame in response to a frame error
	`FRMERR<CR><LF>`
			Required

		`Connectiontype` The Connectiontype parameter is used to designate a connection type. `0: keep-alive` (Transport connection not disconnected after this frame. When PUCC protocols are over OBEX, this value must be set.) `1: single` (Disconnect transport connection after this frame.)	Required
		`FrameNo` The FrameNo parameter is sequential number used to distinguish frames. This parameter is set at the default value at 0 and final value is 2147483647. If a message is divided, "-SeqNo/WholePacketCounts" is given to the end of FrameNo. The frame is specified by changing only the part of "- SeqNo/WholePacketCounts". The position of a frame is specified by "-SeqNo/WholePacketCounts".	Required
		`Size` The Size parameter is used to designate size of payload (without header and trailer) by the octet number.	Required
		`DestNodeID` The DestNodeID parameter is used to designate destination node ID. If destination node ID is not fixed, "*" is specified.	Required
		`SrcNodeID` The SrcNodeID parameter is used to designate source node ID.	Required
		`CommunityID` The CommunityID parameter is used to distinguish which community the message belongs to.	Optional

Payload	C) Frame of normal message
	`(MIME entity header) <CR><LF> [PUCC message]`
	The MIME entity header is optional. The Payload has one PUCC message.
	D) Frame to response to a frame error
	`[Header of P2PFRM]`
	The Payload has one header of P2PFRM which is regarded as error.
			Required

Trailer	`FRMEND<CR><LF>`		Required

Osano 2008/5/26 14:34
書式変更: 箇条書きと段落番号

Osano 2008/5/26 14:34
書式変更: 箇条書きと段落番号

The following shows a sample P2PFRM frame.

```
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:X

<Core xmlns=" Namespace of PUCC Core Protocol" >
 //omitted
</Core>
FRMEND
```

Figure 32 A sample of P2PFRM frame

The following shows a sample FRMERR frame.

```
FRMERR
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:X
FRMEND
```

Figure 33 A sample of FRMERR frame

7.5.2. Connection Type

When PUCC protocols are used over OBEX, the ConnectionType value is 0 (keep-alive) .

7.5.3. Transport layer address

The transport layer address of OBX binding uses the URL format. It shall be assumed that the OBEX client has been already notified of the URL of the OBEX server by the TransportAdrdress element in the Resource Infromation, etc.

When IrDA is used for a lower layer of OBEX, the URL of the OBEX client shall use the value of obex:/irclient and the URL of the OBEX server shall use the value of obex:/irserver.

7.5.4. OBEX Header

The OBEX header and setting values that OBEX transport uses are shown below.

Table 34 OBEX Header

OBEX Header name	Description	Value when conveying PUCC protocol messages
Name	Object name	P2PFRM
Type	Object type	text/x-p2pfrm
Time	Object's UTC time of last modification in ISO8601 format.	The time the message is sent
Body	Object	A fragment of P2PFRM frame or FRMERR frame.
End of Body	Last chunk of the object	The last fragment of P2PFRM frame or FRMERR frame.

7.5.5. OBEX role

OBEX is a client –server model protocol, which consists of an OBEX client and an OBEX server.

In OBEX transport binding, one node becomes the OBEX client and the other node becomes the OBEX server.

The OBEX client establishes a session with the OBEX server, using the CONNECT operation.

The OBEX client periodically performs a GET operation and polls the OBEX server for any new messages to be sent to the OBEX client. When the OBEX client has any message to the OBEX server, the OBEX client uses the OBEX PUT operation to transmit the message to the OBEX server. The OBEX client cannot start another OBEX GET operation as long as an OBEX PUT operation is in progress. The OBEX client must wait until the current OBEX PUT operation is over to perform a new OBEX GET operation.

The OBEX server constantly awaits a GET operation from the OBEX client.

When the OBEX server has a message to be sent to the OBEX client, the OBEX server returns a response which contains that message in response to the OBEX GET operation it has received. When the OBEX server has no new message to the OBEX client, the OBEX server transmits a response with an empty message to the OBEX client. When the OBEX client performs a PUT operation to send a message, the OBEX server receives the message and sends a response to the OBEX client.

An OBEX session can be released by either the OBEX client or the OBEX server. This specification does not specify the values for the polling cycle time.

A sample sequence is shown below.

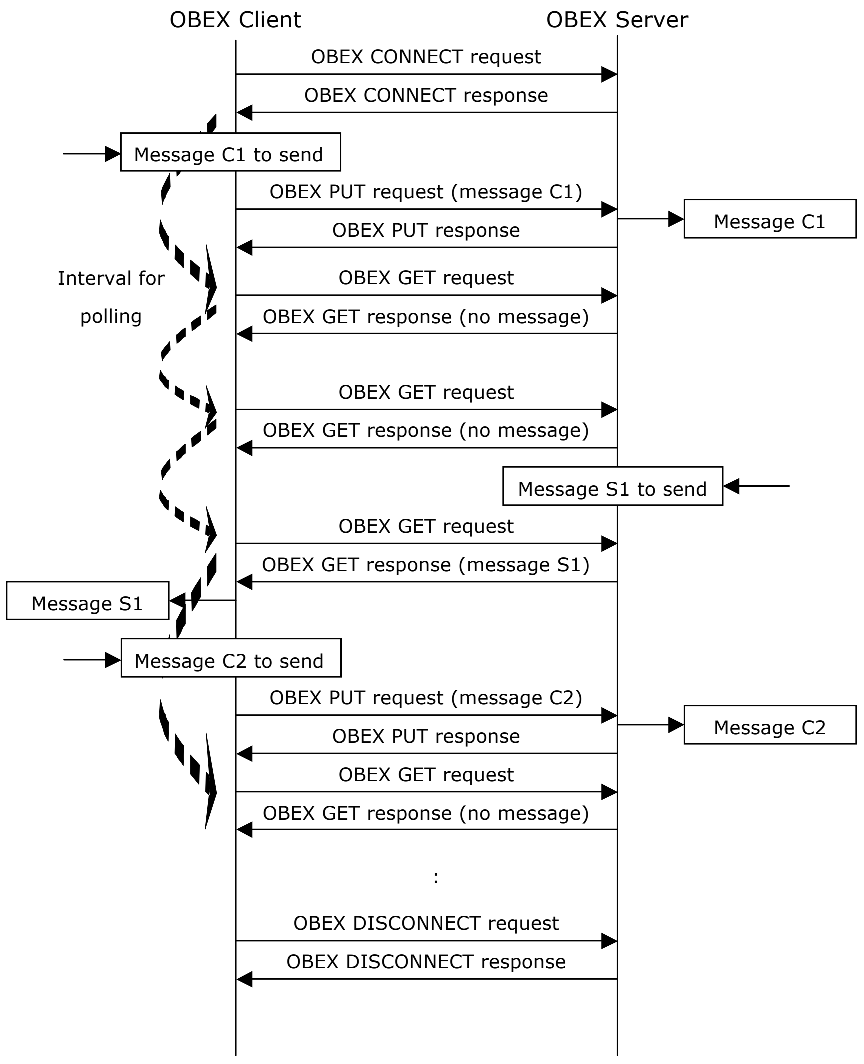


Figure 35 A sample of OBEX protocol sequence

7.6. HTTP Transport

7.6.1. Frame

This section describes the transport binding of the PUCC Protocols over HTTP. All PUCC messages are encapsulated by the following frame when the PUCC Protocol is used over HTTP.

Table 2: The format of the frame for binding of PUCC Protocols over HTTP

	Description	Status
Header	There are 5 kinds of header. A) Frame of normal message `P2PFRM<SP>Connectiontype<SP>FrameNo(- SeqNo/WholePacketCounts)<SP>Size<SP>DestNodeID<SP>SrcNodeID (<SP>CommunityID) <CR><LF>`	Required
	`Connectiontype` The Connectiontype parameter is used to designate a connection type. `0: keep-alive` (Transport connection not disconnected after this frame.) `1: single` (Disconnect transport connection after this frame. When PUCC protocols are over HTTP, this value must be set.)	Required
	`FrameNo` The FrameNo parameter is sequential number used to distinguish frames. This parameter is set at the default value at 0 and final value is 2147483647. If a message is divided, "-SeqNo/WholePacketCounts" is given to the end of FrameNo. The frame is specified by changing only the part of "- SeqNo/WholePacketCounts". The position of a frame is specified by "-SeqNo/WholePacketCounts". iAppli(Doja Platform) sets a limit to the maximum length of one HTTP body.When the frame length exceeds the specified maximum length, segmentation and reassembly of P2PFRM is performed.	Required
	`Size` The Size parameter is used to designate size of payload (without header and trailer) by the octet number.	Required
	`DestNodeID` The DestNodeID parameter is used to designate destination node ID. If destination node ID is not fixed, "*" is specified.	Required
	`SrcNodeID` The SrcNodeID parameter is used to designate source node ID.	Required
	`CommunityID` The CommunityID parameter is used to distinguish which community the message belongs to.	Optional

	B) Frame in response to a frame error
	`FRMERR<CR><LF>`
	
	C) Frame in response to a normal message.
	`ACKFRM<CR><LF>`
	
	D) Frame for polling by HTTP POST request.
	`POLFRM<CR><LF>`
	
	E) Frame representing no message in response to a polling frame.
	`NOMSGFRM<CR><LF>`	

Payload	A) Frame of normal message
	`(MIME entity header) <CR><LF> [PUCC message]`
	The MIME entity header is optional. The Payload has one PUCC message.
	
	B) Frame to response to a frame error
	`[Header of P2PFRM]`
	The Payload has one header of P2PFRM which is regarded as error.
	
	C) Frame in response to a normal message.
	Payload is not defined and not used.
		Optional

	D) Frame for polling. Payload is not defined and not used. E) Frame representing no message left in response to a polling frame. Payload is not defined and not used.	
Trailer	A) Frame of normal message `FRMEND<CR><LF>` B) Frame to response to a frame error `FRMEND<CR><LF>` C) F rame in response to a normal message. Trailer is not defined and not used. D) Frame for polling. Trailer is not defined and not used. E) Frame representing no message left in response to a polling frame. Trailer is not defined and not used.	Optional

kbt 20150501 2016/3/23 16:04
書式変更: インデント: 最初の行: 1 字

The following shows a sample P2PFRM frame.

```
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:X

<Core xmlns=" Namespace of PUCC Core Protocol" >
 //omitted
</Core>
FRMEND
```

Figure 36 A sample of P2PFRM frame

The following shows a sample FRMERR frame.

```
FRMERR
P2PFRM 0 0-1/3 702 965841a8-aa56-8746-b456c458450 113541a8-bb56-8746-b456c457581 urn:pucc:community:X
FRMEND
```

Figure 37 A sample of FRMERR frame

The following shows a sample ACKFRM frame.

```
ACKFRM
```

Figure 38 A sample of ACKFRM frame

The following shows a sample POLFRM frame.

```
POLFRM
```

Figure 39 A sample of POLFRM frame

The following shows a sample NOMSGFRM frame.

```
NOMSGFRM
```

Figure 40 A sample of NOMSGFRM frame

7.6.2. Connection Type

When a PUCC protocol is used over HTTP, the ConnectionType value is 1 (single) .

It does not mean that persistent connections specified by HTTP/1.1 should not be used. The ConnectionType in the HTTP binding does not set the connection type of HTTP in the transport layer (TCP) but sets the connection type of the PUCC protocol in the transport layer (HTTP). HTTP does not have a concept of connection type (since it is a session layer protocol and at the same time a state less protocol.) Therefore, the ConnectionType is set to 1:single.

This process is not involved in imposing limits on the configuration of HTTP Connection headers or Keep-Alive headers.

7.6.3. Mapping of PUCC connection with HTTP session

HTTP is a client –server model protocol. Therefore, one PUCC node plays an HTTP client role, while the other PUCC node plays an HTTP server role in communications. Two cases of HTTP binding can be specified. One case is where both PUCC nodes can play the role of an HTTP client and a HTTP server simultaneously. The other case is that one PUCC node can only play the role of either an HTTP client or an HTTP server. In the former case, one PUCC connection is realized by two HTTP sessions: one to send and the other to receive PUCC protocol messages. In the latter case, one PUCC connection is realized by a single HTTP session.

A) A case where both PUCC nodes support the role of HTTP Client and HTTP Server

In this case, a P2PFRM frame which has encapsulated a PUCC protocol message performs transmission on one HTTP session and reception on the other HTTP session in an asynchronous manner. A P2PFRM frame which has encapsulated a PUCC protocol message is configured in the body part of an HTTP POST request message to send and receive data in each transmission direction. Note that a P2PFRM frame which has encapsulated a PUCC response message may be also configured in the body part of an HTTP POST response message if that is possible. An ACKFRM frames are used for other HTTP POST response messages.

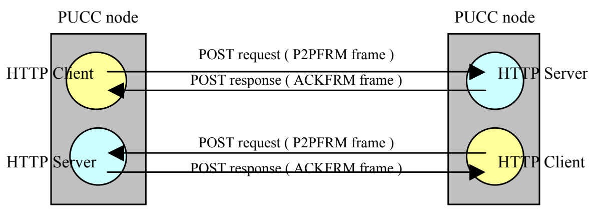

PUCC node
PUCC node
HTTP Client
HTTP Server
POST request (P2PFRM frame)
POST response (ACKFRM frame)
HTTP Server
HTTP Client
POST request (P2PFRM frame)
POST response (ACKFRM frame)

Figure 41 In case PUCC nodes support both HTTP client and HTTP server role

The mapping of a PUCC protocol message and an HTTP message in the above case is shown below.

Table 42 Mapping of PUCC protocol message to HTTP message

MsgType of the PUCC protocol message	send / receive	HTTP message
"Request"	Send	HTTP POST request
	Receive	HTTP POST request
"Response"	Send	HTTP POST request or HTTP POST response
	Receive	HTTP POST request or HTTP POST response
"Advertise"	Send	HTTP POST request
	Receive	HTTP POST request

The sequence is shown below..

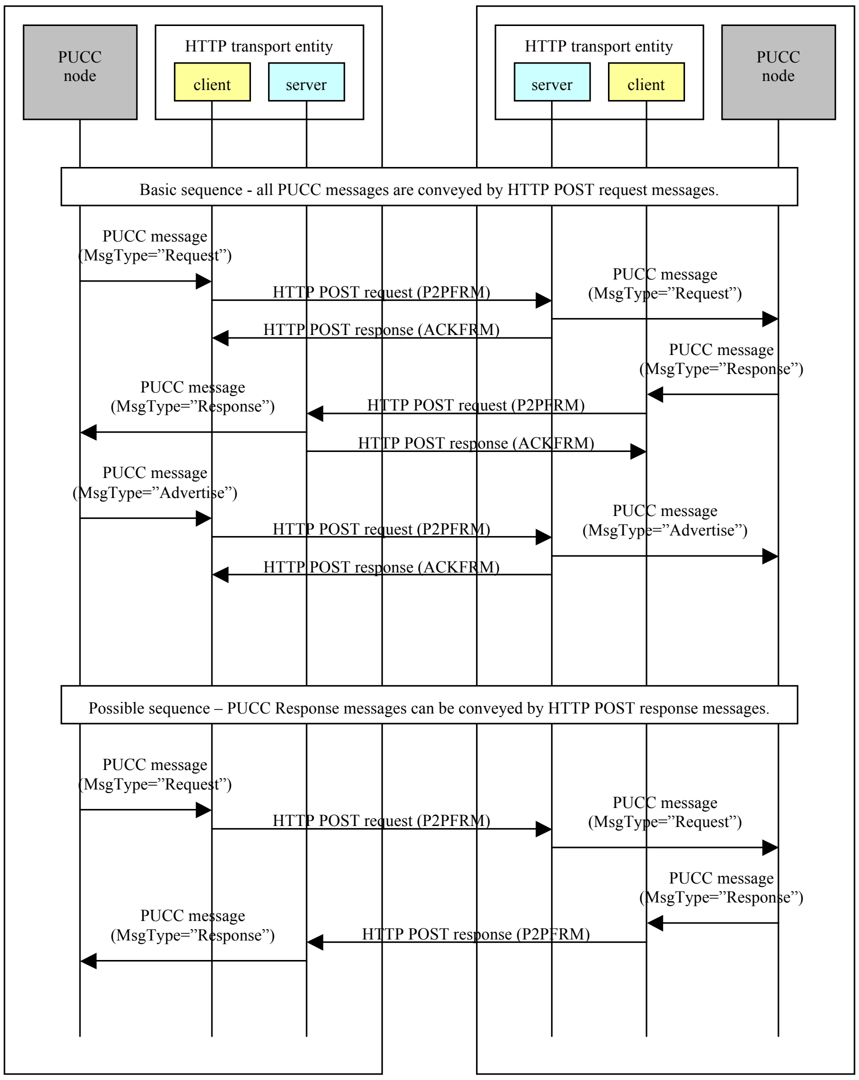

PUCC node
HTTP transport entity
client
server
HTTP transport entity
server
client
PUCC node
Basic sequence - all PUCC messages are conveyed by HTTP POST request messages.
PUCC message (MsgType="Request")
HTTP POST request (P2PFRM)
PUCC message (MsgType="Request")
HTTP POST response (ACKFRM)
PUCC message (MsgType="Response")
PUCC message (MsgType="Response")
HTTP POST request (P2PFRM)
HTTP POST response (ACKFRM)
PUCC message (MsgType="Advertise")
HTTP POST request (P2PFRM)
PUCC message (MsgType="Advertise")
HTTP POST response (ACKFRM)
Possible sequence – PUCC Response messages can be conveyed by HTTP POST response messages.
PUCC message (MsgType="Request")
HTTP POST request (P2PFRM)
PUCC message (MsgType="Request")
PUCC message (MsgType="Response")
PUCC message (MsgType="Response")
HTTP POST response (P2PFRM)

Figure 43 Sequence of sample PUCC protocol message exchange over HTTP between PUCC nodes which support both HTTP client role and HTTP server role

B) A case where one of the PUCC nodes supports the role of either an HTTP Client or HTTP Server

In this case, a P2PFRM frame which has encapsulated a PUCC protocol message performs transmission and reception on one HTTP session in a synchronous manner. A P2PFRM frame which has encapsulated a PUCC protocol message is configured in the body part of an HTTP POST request message in one direction and in the body part of an HTTP POST response message in the other direction.

The node which plays the HTTP server role cannot push HTTP messages to the node which plays the HTTP client role. Therefore, a smart pull (polling) is periodically performed from the node playing the HTTP client role to the one playing the HTTP server role if any PUCC protocol message with the MsgType of "Request" or "Advertise" is sent from the node playing the HTTP server role to the one playing the HTTP client role.
A POLFRM frame indicating polling is configured in the body part of the HTTP POST request message to be sent for the polling from the PUCC node playing the HTTP client role to the PUCC node playing the HTTP server role.

If, upon receiving the polling, the PUCC node playing the HTTP server role has any PUCC protocol message to be sent to the PUCC node playing the HTTP client role, a P2PFRM frame is configured in the body part of the HTTP POST response message. If there is not PUCC protocol message to be sent from the PUCC node playing the HTTP server role to the PUCC node playing the HTTP client role, a NOMSGRFM frame is configured in the body part of the HTTP POST response message to indicate that there is no PUCC protocol message in it. This specification does not specify any value of polling cycle time.

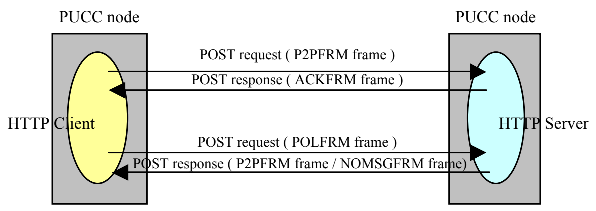

PUCC node
PUCC node
POST request (P2PFRM frame)
POST response (ACKFRM frame)
HTTP Client
HTTP Server
POST request (POLFRM frame)
POST response (P2PFRM frame / NOMSGFRM frame)

Figure 44 In case PUCC nodes support either HTTP client or HTTP server role

The mapping of PUCC protocol messages and HTTP messages in the above case is shown below.

Table 45 Mapping of PUCC protocol message to HTTP message

MsgType of the PUCC protocol message	send / receive	HTTP message
“Request”	Send	HTTP POST request
	Receive	HTTP POST response (by polling)
“Response”	Send	HTTP POST request
	Receive	HTTP POST response (by polling or not)
“Advertise”	Send	HTTP POST request
	Receive	HTTP POST response (by polling)

The sequence is shown below.

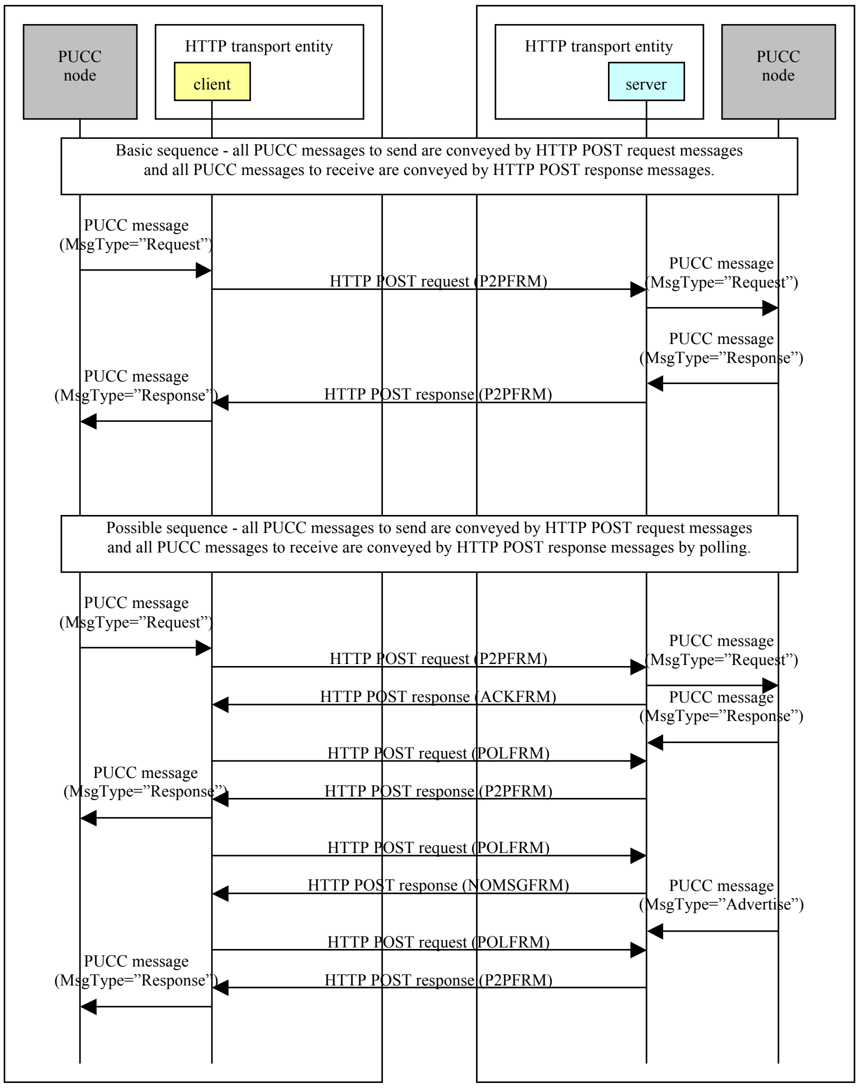

PUCC node
HTTP transport entity
client
HTTP transport entity
server
PUCC node
Basic sequence - all PUCC messages to send are conveyed by HTTP POST request messages and all PUCC messages to receive are conveyed by HTTP POST response messages.
PUCC message (MsgType="Request")
HTTP POST request (P2PFRM)
PUCC message (MsgType="Request")
PUCC message (MsgType="Response")
PUCC message (MsgType="Response")
HTTP POST response (P2PFRM)
Possible sequence - all PUCC messages to send are conveyed by HTTP POST request messages and all PUCC messages to receive are conveyed by HTTP POST response messages by polling.
PUCC message (MsgType="Request")
HTTP POST request (P2PFRM)
PUCC message (MsgType="Request")
HTTP POST response (ACKFRM)
PUCC message (MsgType="Response")
HTTP POST request (POLFRM)
PUCC message (MsgType="Response")
HTTP POST response (P2PFRM)
HTTP POST request (POLFRM)
HTTP POST response (NOMSGFRM)
PUCC message (MsgType="Advertise")
HTTP POST request (POLFRM)
PUCC message (MsgType="Response")
HTTP POST response (P2PFRM)

Figure 46 Sequence of sample PUCC protocol message exchange over HTTP between PUCC node which supports HTTP client role and PUCC node which supports HTTP server role

7.6.4. Transport layer address

A transport layer address of the HTTP binding is written in the URL format. This specification does not define any URL for the PUCC node supporting the HTTP server role. It shall be assumed that the PUCC node supporting the HTTP client role has been already notified of the URL of the PUCC node supporting the HTTP sever role by the TransportAddress element of the Resource Information, etc.

The address of the PUCC node supporting the HTTP client role is set and fixed to a special address of “http:/httpclient” in the TransportAddress element of the Resource Information. This URL indicates that the node to which this URL is set is supposed to behave as an HTTP client.

7.6.5. HTTP version

This specification does not specify any HTTP version number it supports.

7.6.6. HTTP status code

When any frame specified in this specification is configured in an HTTP POST response, the value of the status code of the HTTP POST response shall be “200 OK.” This specification does not specify any HTTP POST response message if any other value is configured to its status code.

7.6.7. HTTP header

a) Content-Type

The Content-Type header of any frame specified in this specification shall be set to “text/x-p2pfrm.”

Note that the frame can form MIME multipart content together with any other content. In such a case, the Content-Type header of the multipart content shall be set to the multipart content type such as multipart/form data, and the Content-Type header of the frame part shall be set to ”text/x-p2pfrm.”

b) Content-Length

The Content-Length header of any frame specified in this specification shall be set to the length of the frame.

Note that the frame can form MIME multipart content together with any other content. In such a case, the Content-Length header of the multipart content shall be set to the total length of the multipart content, and the Content-Length header of the frame part shall be set to the length of the frame.

c) Content-Encoding

The Content-Encoding header of any frame specified in this specification shall not be set to “chunked.”

d) Host

The Host header of an HTTP POST request for HTTP 1.1 shall be set to the FQDN of the server (or the port number when any port other than the Well-Know port is designated).

Appendix A: Version History

Document number	Date	Note
PUCC Basic Protocol – light profile	30 Sep, 2007	Version 1.0
PUCC Basic Protocol – light profile	31 Mar, 2009	Version 2.0

Appendix B: First Peer Discovery

To connect to a PUCC network, a PUCC node must first locate the first (nearest) PUCC node to which it then connects.

In the light PUCC architecture, the joining node sends a Lookfor message around the physical network. As shown in Fig. A-1, when an existing PUCC node receives the Lookfor message, it replies to the joining node with a LookforResponse message. The joining node connects to the PUCC network according to LookforResponse messages received. The Lookfor message is a special connectionless message. For PUCC over IP network, the Lookfor message is realized as an IP multicast message. The LookforResponse message is also a connectionless message. In a PUCC over IP network, the LookforResponse message is realized as an IP multicast message or an IP unicast message.

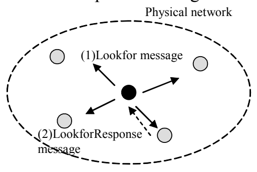


Figure 47. First peer discovery in light PUCC architecture

The sequence of first peer discovery in the light PUCC architecture is shown in Fig A-2.

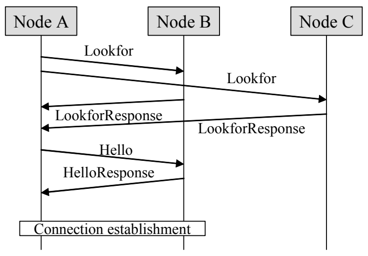


Figure 48. Sequence of the first peer discovery in light PUCC architecture

Appendix C: NAT Transmission

This section describes consideration about how to make a PUCC node in LAN participate in a PUCC network in WAN using an UPnP-enabled IP router.

A) What's UPnP?

UPnP is a mechanism that makes various devices connected to IP network, recognize one another and interconnect seamlessly without complicated settings by users.

The most common function of an UPnP-enabled IP router is automatic NAT setting function cooperated with UPnP-enabled applications. The windows messenger bundled with Windows XP is one of the famous UPnP-enabled applications.

UPnP-enabled devices and UPnP-enabled applications consist of device, service and control point.

- **Device**

"Device" means each UPnP-enabled device and UPnP-enabled application. It is a container of service components. Multiple devices can compose one device. (Combo device)

- **Service**

"Service" means functions provided by UPnP-enabled devices and applications. The way to apply and control the services is called "Action".

For instance, a clock device provides a timer service. The timer service has a time setting action and a time acquisition action and so on.

- **Control Point**

It is a controller to detect devices and use services.

It can be embedded in UPnP-enabled devices and applications, and also can independently exist.

Information of device, service and action is defined using a "Device description document" based on XML. The device has HTTP server and responses a request to get a device description document from the control point.

Following protocol is used for communications between the control point and devices.

- **HTTP/HTTPU/HTTPMU**

All protocol used in UPnP system is based on HTTP. HTTPU is HTTP using UDP. HTTPMU is HTTP using UDP multicast.

- **SSDP (Simple Service Discovery Protocol)**

This protocol is used for control point to detect services.

The control point sends a request message (SSDP search request) to search devices and services using HTTPMU.

The devices always listens UDP port. When it receives an appropriate request message, it replies a response message to the control point using HTTPU.

Conversely, when the device connects to a network, it sends a SSDP presence announce using HTTPMU to notify the control point of its own services.

・ **GENA (Generic Event Notification Architecture)**

GENA is defined as function to send and receive a notification message using HTTP over TCP/IP and multicast UDP. It is a mechanism to notify the control point of service state change by the devices.

・ **SOAP (Simple Object Access Protocol)**

The control point sends a message to devices to use services provided by the devices using SOAP. The device uses SOAP to send a response to the control point, too.

B) IGD (InternetGatewayDevice)

The specification of UPnP-enabled router is provided in "InternetGatewayDevice (IGD) V1.0"

The specification is available in UPnP forum（http://www.upnp.org/）.

IGD provides "Layer3Forwarding Service. And IDG has one or more "WANDevice" and one or more "LAN Device". These devices respectively provide services and "WANDevice" has multiple "WANConnectionDevices".

IDG is not for user or device authentication mechanism and access control mechanism.

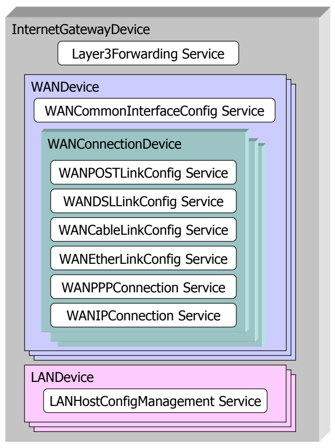


Figure 49. Devices and services of InternetGatewayDevice

C) Application of UPnP to PUCC Platform (Light PUCC network)

When internal and external PUCC network are light PUCC networks, an internal particular light PUCC node has to have multiple IP address (Global and local). The gateway-like light PUCC node has to control both local and global IP addresses and choose them according to target nodes. In current PUCC Basic Protocol – light profile specification, a node can not distinguish between an external PUCC node and an internal PUCC node. It is needed to add such as information of netmask to distinguish the networks.

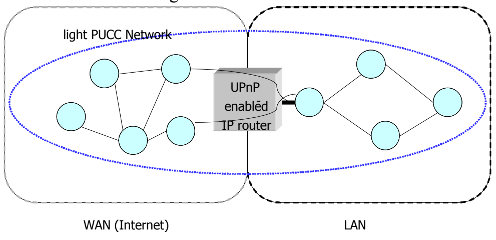


Figure 50. Light PUCC network using an UPnP-enabled router

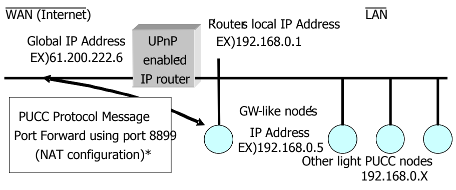

WAN (Internet)
LAN
Global IP Address EX)61.200.222.6
UPnP enabled IP router
Routes local IP Address EX)192.168.0.1
PUCC Protocol Message Port Forward using port 8899 (NAT configuration)*
GW-like nodes
IP Address EX)192.168.0.5
Other light PUCC nodes 192.168.0.X

Figure 51. Relationship between a light PUCC node and an UPnP-enabled router

The gateway-like light PUCC node works as a control point of UPnP and acquires global IP address of the router and sets up the port forwarding.

Appendix D: Use cases

A) Distributed Metadata Search Application

The PUCC network is more scalable and offers better management than the client-server model. The PUCC network can be used for locating distributed data in the ubiquitous communication environment. Distributed Metadata Search Applications can be developed with PUCC Protocols.

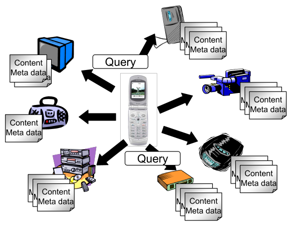


Figure 52. Distributed Metadata Search in the ubiquitous communication environment

B) Mobile Push Service

The PUCC network is adaptable for ad-hoc communication. The PUCC network can be used for advertising push service based on the user's location on the street.

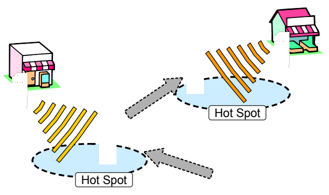


Figure 53. Advertising Push Service on the street

C) Small Personal Network Platform

The PUCC network can construct a confidential network. The PUCC network can be used as a Home appliance network and a personal area network and so on.

Figure 54. Personal Area Network

Appendix E: DTD for PUCC Basic Communication Protocol

A) DTD for Hello message

```
<?xml version="1.0"?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, SessionID?, MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ELEMENT Destination (Target)>
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT MsgBody (Hello)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">
<!ELEMENT Hello (InitiatorUserID?, ResponderUserID?)>
<!ATTLIST Hello xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com”
<!ELEMENT InitiatorUserID (#PCDATA)>
<!ELEMENT ResponderUserID (#PCDATA)>
]>
```

B) DTD for HelloResponse message

```
<?xml version="1.0"?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, SessionID?, MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT ReplyID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ELEMENT Destination (Target)>
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT MsgBody (HelloResponse)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">
<!ELEMENT HelloResponse (Result, Reason?)>
<!ATTLIST HelloResponse CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">
<!-- Reason element is required when Result value (#PCDATA) is Failure.
     However, Reason element should be absent when Result value (#PCDATA) is Success. -->
<!ELEMENT Result (#PCDATA)>
<!ELEMENT Reason (#PCDATA)>
]>
```

C) DTD for Bye message

```
<?xml version="1.0"?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, SessionID?, MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED " http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ELEMENT Destination (Target)>
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT MsgBody (Bye)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">
<!ELEMENT Bye EMPTY>
<!ATTLIST Bye xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">
]>
```

D) DTD for ResourceInformationRequest message

```
<?xml version="1.0" ?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination?, TraceRoute, HopCount?, SessionID?,
MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ENTITY % Route.class "(Route | Target)*" >
<!ELEMENT Route %Route.class;>
<!ATTLIST Route Node CDATA #REQUIRED>
<!ELEMENT Destination %Route.class;>
<!-- Destination element is required when ComType (#PCDATA) is Unicast.
     However, Destination element should be absent when ComType value (#PCDATA) is Broadcast. -->
<!ELEMENT TraceRoute %Route.class;>
<!-- Route element is required within Destination element if the distance from Source node to Target node is
     more than 1 hop. However, Route element should be absent within Destination element if the distance
     from Source node to Target node is only 1 hop.
     Route element is always required within TraceRoute element. -->
<!ELEMENT HopCount (#PCDATA)>
<!-- HopCount element is required when ComType value (#PCDATA) is Broadcast and hop count is finite.
     HopCount element should be absent when ComType value (#PCDATA) is Broadcast and hop count is infinite.
     HopCount element always should be absent when ComType value (#PCDATA) is Unicast. -->
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT MsgBody (ResourceInformationRequest)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">
<!ELEMENT ResourceInformationRequest EMPTY>
<!ATTLIST ResourceInformationRequest xmlns CDATA #FIXED “http://www.pucc.jp/2007/09/basic_com”>
]>
```

E) DTD for ResourceInformationResponse message

```
<?xml version="1.0" ?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, TraceRoute, SessionID?,
MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT ReplyID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ENTITY % Route.class "(Route? | Target)" >
<!ELEMENT Route %Route.class;>
<!ATTLIST Route Node CDATA #REQUIRED>
<!ELEMENT Destination %Route.class;>
<!ELEMENT TraceRoute %Route.class;>
<!-- Route element is required within Destination element if the distance from Source node to Target node is
    more than 1 hop. However, Route element should be absent within Destination element if the distance
    from Source node to Target node is only 1 hop.
    Route element is always required within TraceRoute element. -->
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT MsgBody (ResourceInformationResponse)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">
<!ELEMENT ResourceInformationResponse (ResourceData+)>
<!ATTLIST ResourceInformationResponse xmlns
CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com”>
<!ELEMENT ResourceData (OwnNodeID, OwnedApplication?, ConnectionCapability, ConnectedNode, TransportAddress+)>
<!ELEMENT OwnNodeID (#PCDATA)>
<!ELEMENT OwnedApplication (Application*)>
<!ELEMENT Application (#PCDATA)>
<!ELEMENT ConnectionCapability (#PCDATA)>
<!ELEMENT ConnectedNode (Node*)>
<!ELEMENT Node (TransportAddress+, ConnectionCapability?, RTT?)>
<!ATTLIST Node ID CDATA #REQUIRED>
<!ELEMENT TransportAddress (#PCDATA)>
<!ATTLIST TransportAddress protocol CDATA #REQUIRED>
```

```
<!ATTLIST TransportAddress type CDATA #REQUIRED>
<!ELEMENT RTT (#PCDATA)>
]>
```

F) DTD for ResourceInformationAdvertise message

```
<?xml version="1.0" ?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination?, TraceRoute, HopCount?, SessionID?,
MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ENTITY % Route.class "(Route? | Target)" >
<!ELEMENT Route %Route.class;>
<!ATTLIST Route Node CDATA #REQUIRED>
<!ELEMENT Destination %Route.class;>
<!-- Destination element is required when ComType (#PCDATA) is Unicast.
     However, Destination element should be absent when ComType value (#PCDATA) is Broadcast. -->
<!ELEMENT TraceRoute %Route.class;>
<!-- Route element is required within Destination element if the distance from Source node to Target node is
     more than 1 hop. However, Route element should be absent within Destination element if the distance
     from Source node to Target node is only 1 hop.
     Route element is always required within TraceRoute element. -->
<!ELEMENT HopCount (#PCDATA)>
<!-- HopCount element is required when ComType value (#PCDATA) is Broadcast and hop count is finite.
     HopCount element should be absent when ComType value (#PCDATA) is Broadcast and hop count is infinite.
     HopCount element always should be absent when ComType value (#PCDATA) is Unicast. -->
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT MsgBody (ResourceInformationAdvertise)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/basic_com">
<!ELEMENT ResourceInformationAdvertise (ResourceData*)>
<!ATTLIST ResourceInformationAdvertise xmlns CDATA #FIXED “http://www.pucc.jp/2007/09/basic_com”>
<!ELEMENT ResourceData (OwnNodeID, OwnedApplication?, ConnectionCapability, ConnectedNode, TransportAddress+)>
<!ELEMENT OwnNodeID (#PCDATA)>
<!ELEMENT OwnedApplication (Application*)>
<!ELEMENT Application (#PCDATA)>
<!ELEMENT ConnectionCapability (#PCDATA)>
<!ELEMENT ConnectedNode (Node*)>
<!ELEMENT Node (TransportAddress+, ConnectionCapability?, RTT?)>
<!ATTLIST Node ID CDATA #REQUIRED>
<!ELEMENT TransportAddress (#PCDATA)>
<!ATTLIST TransportAddress protocol CDATA #REQUIRED>
<!ATTLIST TransportAddress type CDATA #REQUIRED>
<!ELEMENT RTT (#PCDATA)>
]>
```

Appendix I: DTD for PUCC Control Message Protocol

A) DTD for ErrorReport message

```
<?xml version="1.0" ?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination, TraceRoute?, SessionID?, MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ENTITY % Route.class "(Route | Target)*" >
<!ELEMENT Route %Route.class;>
<!ATTLIST Route Node CDATA #REQUIRED>
<!ELEMENT Destination %Route.class;>
<!-- Destination element is required when ComType (#PCDATA) is Unicast.
    However, Destination element should be absent when ComType value (#PCDATA) is Broadcast. -->
```

```
<!ELEMENT TraceRoute %Route.class;>
<!-- Route element is required within Destination element if the distance from Source node to Target node is
     more than 1 hop. However, Route element should be absent within Destination element if the distance
     from Source node to Target node is only 1 hop.
     Route element is always required within TraceRoute element. -->
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT MsgBody (ErrorReport)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ELEMENT ErrorReport (ErrorMsgID, Error)>
<!ATTLIST ErrorReport xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ELEMENT ErrorMsgID (#PCDATA)>
<!ELEMENT Error (#PCDATA)>
```

B) DTD for Diagnose message

```
<?xml version="1.0" ?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, Destination?, TraceRoute, HopCount?, SessionID?,
MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ENTITY % Route.class "(Route | Target)*" >
<!ELEMENT Route %Route.class;>
<!ATTLIST Route Node CDATA #REQUIRED>
<!ELEMENT Destination %Route.class;>
<!-- Destination element is required when ComType (#PCDATA) is Unicast.
     However, Destination element should be absent when ComType value (#PCDATA) is Broadcast. -->
<!ELEMENT TraceRoute %Route.class;>
<!-- Route element is required within Destination element if the distance from Source node to Target node is
     more than 1 hop. However, Route element should be absent within Destination element if the distance
```

```
        from Source node to Target node is only 1 hop.
        Route element is always required within TraceRoute element. -->
<!ELEMENT HopCount (#PCDATA)>
<!-- HopCount element is required when ComType value (#PCDATA) is Broadcast and hop count is finite.
        HopCount element should be absent when ComType value (#PCDATA) is Broadcast and hop count is infinite.
        HopCount element always should be absent when ComType value (#PCDATA) is Unicast. -->
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT MsgBody (Diagnose)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ELEMENT Diagnose (DiagnoseData?, DiagnoseDestination?)>
<!ATTLIST Diagnose xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ELEMENT DiagnoseData (#PCDATA)>
<!ELEMENT DiagnoseDestination (#PCDATA)>
<!ATTLIST DiagnoseDestination type (”NodeID” | ”ApplicationID”) #REQUIRED>
]>
```

C) DTD for DiagnoseResponse message

```
<?xml version="1.0" ?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, TraceRoute, SessionID?,
MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT ReplyID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ENTITY % Route.class "(Route | Target)*" >
<!ELEMENT Route %Route.class;>
<!ATTLIST Route Node CDATA #REQUIRED>
<!ELEMENT Destination %Route.class;>
<!ELEMENT TraceRoute %Route.class;>
<!-- Route element is required within Destination element if the distance from Source node to Target node is
        more than 1 hop. However, Route element should be absent within Destination element if the distance
        from Source node to Target node is only 1 hop.
        Route element is always required within TraceRoute element. -->
<!ELEMENT SessionID (#PCDATA)>
<!ELEMENT MsgBody (DiagnoseResponse)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ELEMENT DiagnoseResponse (DiagnoseData)>
<!ATTLIST DiagnoseResponse xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ELEMENT DiagnoseData (#PCDATA)>
]>
```

D) DTD for Lookfor message

```
<?xml version="1.0" ?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, MsgType, CommunityID?, Source, MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT MsgBody (Lookfor)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ELEMENT Lookfor (Condition?)>
<!ATTLIST Lookfor xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ATTLIST Lookfor count CDATA #IMPLIED>
<!ELEMENT Condition (#PCDATA)>
<!ATTLIST Condition protocol CDATA #REQUIRED>
]>
```

E) DTD for LookforResponse message

```
<?xml version="1.0" ?>
<!DOCTYPE Core [
<!ELEMENT Core (ComType, MsgID, ReplyID, MsgType, CommunityID?, Source, Destination, MsgBody)>
<!ATTLIST Core xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/core">
<!ELEMENT ComType (#PCDATA)>
<!ELEMENT MsgID (#PCDATA)>
<!ELEMENT ReplyID (#PCDATA)>
<!ELEMENT MsgType (#PCDATA)>
<!ELEMENT CommunityID (#PCDATA)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT Target (#PCDATA)>
<!ELEMENT Destination (Target)>
<!ELEMENT MsgBody (LookforResponse)>
<!ATTLIST MsgBody protocol CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ELEMENT LookforResponse (Description?, DeviceList?)>
<!ATTLIST LookforResponse xmlns CDATA #FIXED "http://www.pucc.jp/2007/09/ctrl_msg">
<!ELEMENT Description (#PCDATA)>
<!ELEMENT DeviceList (Device+)>
<!ELEMENT Device (URLBase?, Manufacturer?, ManufacturerURL?, ManufactureDate?, ModelDescription?, ModelName?,
ModelNumber?, ModelURL?, SerialNumber?, UDN?, UPC?, IconList?)>
<!ELEMENT URLBase (#PCDATA)>
<!ELEMENT Manufacturer (#PCDATA)>
<!ELEMENT ManufacturerURL (#PCDATA)>
<!ELEMENT ManufactureDate (#PCDATA)>
<!ELEMENT ModelDescription (#PCDATA)>
<!ELEMENT ModelName (#PCDATA)>
<!ELEMENT ModelNumber (#PCDATA)>
<!ELEMENT ModelURL (#PCDATA)>
<!ELEMENT SerialNumber (#PCDATA)>
<!ELEMENT UDN (#PCDATA)>
<!ELEMENT UPC (#PCDATA)>
<!ELEMENT IconList (Icon+)>
<!ELEMENT Icon (Mimetype?, Width?, Height?, Depth?, Url?)>
<!ELEMENT Mimetype (#PCDATA)>
<!ELEMENT Width (#PCDATA)>
<!ELEMENT Height (#PCDATA)>
```

```
<!ELEMENT Depth (#PCDATA)>
<!ELEMENT Url (#PCDATA)>
<!ATTLIST Device type CDATA #REQUIRED>
<!ATTLIST Device id CDATA #IMPLIED>
<!ATTLIST Device name CDATA #IMPLIED>
]>
```

Appendix J: Namespace Definitions

Namespaces are defined with URI.

A) PUCC Core Protocol

http://www.pucc.jp/2007/09/core

xmlns attribute of Core element uses it.

B) PUCC Basic Communication Protocol

http://www.pucc.jp/2007/09/basic_com

xmlns attribute of MsgBody element uses it in PUCC Basic Communication Protocol.

C) PUCC Control Message Protocol

http://www.pucc.jp/2007/09/ctrl_msg

xmlns attribute of MsgBody element uses it in PUCC Control Message Protocol.