[MC-PRCR]:
Peer Channel Custom Resolver Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
08/10/2007	0.1	Major	Initial Availability
09/28/2007	0.2	Minor	Updated the technical content.
10/23/2007	0.3	Minor	Updated the technical content.
11/30/2007	0.3.1	Editorial	Revised and edited the technical content; updated links.

Date	Revision History	Revision Class	Comments
01/25/2008	0.3.2	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

The Peer Channel Custom Resolver Protocol is used for storage and retrieval of **endpoint information** of clients with access to a known service. Clients using the service can store their own endpoint information at the service and obtain endpoint information about other clients in order to establish direction connections between them.

1.1 Glossary

ClientId: Unique string that is used to identify itself to the **resolver service**, as described in section 1.3.1.2.

Endpoint Information: An endpoint address, formatted as a URI, along with a set of IP addresses. Describes the set of addresses on which a node is listening.

Mesh Name: Unique identifier of a collection of **registration records**. The nodes referred to by these **registration records** typically reside in a single mesh, as defined in [MC-PRCH], section 1.1.

PeerNodeAddress: A structure that contains a node's URI and a set of IP addresses on which the client is listening (section 2.2.1.1).

RegistrationId: A unique identifier in the form of a **GUID**, as described in section 1.3.1.3.

Registration Record: A set of **endpoint information** stored at the **resolver service**.

Registration Lifetime: Associated with a particular **registration record**. This is the amount of time during which a **registration record** is guaranteed to be preserved by the **resolver service**. This time starts upon the successful addition of a **registration record** to the **resolver service**. After this time has elapsed, the **resolver service** may choose to delete the corresponding **registration record** (section 3).

Resolver Client: Application endpoint that uses a **resolver service** to publish and/or obtain **endpoint information**.

Resolver Service: Application implementing this protocol that stores and distributes **endpoint information**.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MC-NMF] Microsoft Corporation, ".NET Message Framing Protocol Specification", October 2007.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[SOAP1.2/1] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624

[XMLDSig] Bartel, M., Boyer, J., Fox, B., LaMacchia, B., and Simon, E., "XML-Signature Syntax and Processing", February 2002, http://www.w3.org/TR/xmldsig-core/

[XMLSCHEMA] World Wide Web Consortium, "XML Schema", September 2005, http://www.w3.org/2001/XMLSchema

1.2.2 Informative References

[MC-PRCH] Microsoft Corporation, "Peer Channel Protocol Specification", October 2007.

[MS-PNRP] Microsoft Corporation, "Peer Name Resolution Protocol (PNRP) Version 4.0 Specification", July 2007.

1.3 Protocol Overview (Synopsis)

This protocol is intended to be used in conjunction with the Peer Channel Protocol [MC-PRCH] as a peer discovery service.

The Peer Channel Custom Resolver Protocol is a client-server protocol used to register and retrieve client endpoint information at a well-known resolver service. The information that is registered and retrieved is the **PeerNodeAddress** of clients associated with a named mesh. This information can then be used to establish direct connections between these clients. Security for this protocol is configured by a higher-layer protocol or application.

There are two primary roles in the Peer Channel Custom Resolver Protocol:

Resolver service - Stores endpoint information and handles requests to add, delete, or update that information.

Resolver client - Sends requests to the resolver service to add, delete, or update endpoint information, or to request a list of the endpoint information of other clients.

1.3.1 Identifiers

1.3.1.1 Mesh Name

Registration information is associated with a string called a **mesh name**, which is used to group related sets of endpoint information. When **registration records** are created or requested, the corresponding mesh name must be included. Endpoint information is then organized by mesh name at the resolver service. The resolver service guarantees that clients requesting endpoint information corresponding to a particular mesh name will only receive endpoint information that was registered with the same mesh name.

1.3.1.2 ClientId

Each resolver client generates a unique string that is used to identify itself to the resolver service. This ID remains constant throughout the lifetime of the resolver client.

1.3.1.3 RegistrationId

When a new registration record is created at the resolver service, it is assigned a unique RegistrationId in the form of a **GUID**. This RegistrationID remains valid for that registration record until the registration record expires or is unregistered.

1.3.2 Connecting to the Resolver Service

The resolver client is configured with the location of the resolver service and a transport that matches the requirements of the service. All clients need to be configured to use the same transport. The connection used by the custom resolver service and the custom resolver client to communicate is protocol independent.

1.3.3 List of Resolver Service Operations

The Peer Channel Custom Resolver Protocol is initiated by a request from a resolver client (with the sole exception of registration record maintenance; see section 1.3.4). The following operations are supported by the resolver service:

Register: Creates a new registration record at the resolver service and populates it with endpoint information contained in the register request.

Update: Changes the information contained in a specific registration record.

Unregister: Deletes a registration record.

Refresh: Changes the expiry time associated with a registration record (section 1.3.4).

Resolve: Retrieves a collection of registration records associated with a particular mesh name.

GetServiceInfo: Retrieves a Boolean value indicating whether or not a corresponding PeerChannel node will use referrals to initiate neighbor connections.

1.3.4 Registration Record Maintenance

Registration records reside at the custom resolver service and are associated with an expiry time and date. These records are to be deleted or marked as invalid by that expiry time (which is determined by the implementation of the service), unless the resolver client that initially registered that endpoint information sends a request to the service that the record be stored longer.

In the case that the resolver service fails, the resolver clients using the service will attempt to refresh or re-register their information. Once the resolver service is restarted, it will repopulate its data store of registration records upon receiving the refresh or re-registration requests from the resolver clients.

1.4 Relationship to Other Protocols

This protocol is transport agnostic and so may be used in conjunction with a variety of transport protocols (TCP, HTTP, and so on).

Messages used in this protocol are formatted in XML/SOAP ([SOAP1-2.1], [XMLDSig]). This involves the use of .NET Message Framing Protocol Specification ([MC-NMF]) for encoding SOAP messages.

This protocol shares characteristics with and could potentially be substituted for other name resolution systems, such as PNRP ([MS-PNRP]). This protocol is intended for use by the Peer Channel Protocol ([MC-PRCH]) for neighbor discovery when PNRP ([MS-PNRP]) is unavailable.

1.5 Prerequisites/Preconditions

All resolver clients must be configured with the location and transport protocol of the resolver service, as well as any security information, if applicable.

1.6 Applicability Statement

This protocol is intended for use in conjunction with the Peer Channel Protocol ([MC-PRCH]) as a means to enable neighbor discovery. This protocol is suitable for storage and distribution of endpoint information or a mapping service associating endpoint information with mesh names. It is not intended nor is it ideal for use with hierarchal namespaces (for example, DNS).

This protocol does not give any guarantees as to the reliability of endpoint information, longevity of registration records, or availability of the resolver service.

1.7 Versioning and Capability Negotiation

The Peer Channel Custom Resolver Protocol has no version-negotiation or capability-negotiation behavior.

1.8 Vendor-Extensible Fields

The Peer Channel Custom Resolver Protocol does not have any vendor-extensible fields.

1.9 Standards Assignments

No standard assignments have been made for this protocol.

2 Messages

2.1 Transport

The transport is configured under a higher-layer protocol or application. This protocol does not specify or require any particular transport mechanism.<1>

2.2 Message Syntax

Peer Channel Custom Resolver messages MUST be formatted using the SOAP syntax ([SOAP1-2.1]).

2.2.1 Structures Used

2.2.1.1 PeerNodeAddress Structure

The PeerNodeAddress structure MUST contain a URI and one or more IPAddresses. It is intended to be used to describe a PeerChannel endpoint. While in use by this protocol, a PeerNodeAddress structure SHOULD only refer to the local endpoint.

```
<xs:schema
  xmlns:tns="http://schemas.datacontract.org/2004/07/System.Net"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace=http://schemas.datacontract.org/2004/07/System.Net
  xmlns:a="http://www.w3.org/2005/08/addressing/"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified" >
  <xs:import namespace="http://schemas.microsoft.com/2003/10/Serializa
tion/Arrays" />
  <xs:import namespace="http://www.w3.org/2005/08/addressing/" />
    <xs:complexType name="IPAddress">
      <xs:sequence>
        <xs:element name="m_Address" type="xs:unsignedInt" />
        <xs:element name="m_Family" type="xs:string" />
        <xs:element name="m_HashCode" type="xs:unsignedInt" />
        <xs:element name="m_Numbers">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" xmlns:q1="http://schem
as.microsoft.com/2003/10/Serialization/Arrays" ref="q1:unsignedShort"
/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="m_ScopeId" type="xs:unsignedInt" />
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PeerNodeAddress">
      <xs:sequence>
        <xs:element name="EndpointAddress">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="a:Address" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="IPAddresses">
```

```
            <xs:complexType>
              <xs:sequence>
                <xs:element maxOccurs="unbounded" ref="d5p1:IPAddress" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
  </xs:schema>
```

ElementName	Description
Address	Reference to the node's endpoint [URI].
IPAddresses	This MUST contain one or more IPAddress structures.
IPAddress	Describes a complete IPAddress.
IPAddress/m_Address	"0" indicates an IPv6 address. Otherwise, address as an unsigned 32-bit number.
IPAddress/m_Family	Address family of the IPAddress. Acceptable strings are: "Internetwork" if the address is an IPv4 address; "InternetworkV6" if the address is an IPv6 address.
IPAddress/m_HashCode	This value MUST be set to "0". Upon parsing this field from a received message, this element MUST be ignored.
IPAddress/m_Numbers	This element contains the serialized version of the address bytes grouped as 16-bit numbers in big-endian format. For IPv4 addresses, this element MAY contain 0 instances. For IPv6 addresses, this element MUST contain exactly 8 "unsignedShort" subelements.
IPAddress/m_Numbers/unsignedShort	MUST contain a 16-bit number.
IPAddress/m_ScopeId	For an IPv6 address, this element MUST contain the Scope ID of the address. For IPv4 addresses, this element MUST be ignored. All IPAddresses in a single PeerNodeAddress MUST have the same ScopeId.

2.2.1.2 RefreshResult Enumeration

This is sent as part of the RefreshResponse message to indicate the result of an attempt to refresh a registration record. It has the following valid values, formatted as strings:

Enumeration Value	Description
Success	The registration record was found and refreshed.
RegistrationNotFound	The resolver service could not find the registration record.

2.2.2 Resolver Client Messages

2.2.2.1 Register Message

The Register message is sent by the client to register its PeerNodeAddress structure with a custom resolver service.

```
<wsdl:operation name="Register">
  <soap12:operation soapAction="http://schemas.microsoft.com/net/2006/
05/peer/resolver/Register" style="document" />
  <wsdl:input name="RegisterInfo">
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_Regis
ter_Input_policy" />
  <soap12:body use="literal" />
  </wsdl:input>
 <wsdl:output name="RegisterResponseInfo">
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_Regis
ter_output_policy" />
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>

<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/net/2006/05/peer" a
ttributeFormDefault="unqualified" elementFormDefault="qualified" targe
tNamespace="http://schemas.microsoft.com/net/2006/05/peer" xmlns:xs="h
ttp://www.w3.org/2001/XMLSchema">
  <xs:import namespace="http://www.w3.org/2005/08/addressing" />
  <xs:import namespace="http://schemas.datacontract.org/2004/07/System
.Net" />
  <xs:element name="Register">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ClientId" type="xs:string" />
        <xs:element name="MeshId" type="xs:string" />
        <xs:element name="NodeAddress" type="PeerNodeAddress" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Element Name	Description
ClientId	String identifying the resolver client. This MUST be unique and SHOULD be created upon the creation of the resolver client. The ClientId of a particular resolver client SHOULD be valid for the lifetime of that client.
MeshId	Mesh name.
NodeAddress	A PeerNodeAddress structure containing the resolver client's endpoint information.

2.2.2.2 Resolve Message

The Resolve message is sent by the resolver client to discover the PeerNodeAddress structure of other clients.

```
<wsdl:operation name="Resolve">
  <soap12:operation soapAction="http://schemas.microsoft.com/net/2006/
05/peer/resolver/Resolve" style="document" />
  <wsdl:input name="ResolveInfo">
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_Resol
ve_Input_policy" />
  <soap12:body use="literal" />
  </wsdl:input>
  <wsdl:output name="ResolveResponseInfo">
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_Resol
ve_output_policy" />
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/net/2006/05/peer" a
ttributeFormDefault="unqualified" elementFormDefault="qualified" targe
tNamespace="http://schemas.microsoft.com/net/2006/05/peer" xmlns:xs="h
ttp://www.w3.org/2001/XMLSchema">
  <xs:element name="Resolve">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ClientId" type="xs:string" />
        <xs:element name="MaxAddresses" type="xs:unsignedByte" />
        <xs:element name="MeshId" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Element Name	Description
ClientId	String identifying the resolver client. This MUST be unique and SHOULD be created upon the creation of the resolver client. The ClientId of a particular resolver client SHOULD be valid for the lifetime of that client.
MaxAddresses	The maximum number of registration records that should be returned by the corresponding ResolveResponse message. This SHOULD be set to "3".
MeshId	Mesh name.

2.2.2.3 Refresh Message

The Refresh message is sent by a client to refresh its registration record with the resolver service.

```
<wsdl:operation name="Refresh">
  <soap12:operation soapAction="http://schemas.microsoft.com/net/2006/
05/peer/resolver/Refresh" style="document" />
- <wsdl:input name="RefreshInfo">
```

```
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_Refre
sh_Input_policy" />
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="RefreshResponseInfo">
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_Refre
sh_output_policy" />
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/net/2006/05/peer" a
ttributeFormDefault="unqualified" elementFormDefault="qualified" targe
tNamespace="http://schemas.microsoft.com/net/2006/05/peer" xmlns:xs="h
ttp://www.w3.org/2001/XMLSchema">
  <xs:element name="Refresh">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="MeshId" type="xs:string" />
        <xs:element name="RegistrationId" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Element Name	Description
MeshId	Mesh Name.
RegistrationId	String that uniquely identifies the registration record of a particular resolver client. Created by the resolver service as a result of processing a Register message.

2.2.2.4 Update Message

The Update message is sent by the resolver client when some details of its registration record need to be updated (for instance, when its IP addresses have changed).

```
<wsdl:operation name="Update">
  <soap12:operation soapAction="http://schemas.microsoft.com/net/2006/
05/peer/resolver/Update" style="document" />
- <wsdl:input name="UpdateInfo">
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_Updat
e_Input_policy" />
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="RegisterResponseInfo">
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_Updat
e_output_policy" />
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/net/2006/05/peer" a
ttributeFormDefault="unqualified" elementFormDefault="qualified" targe
```

```
tNamespace="http://schemas.microsoft.com/net/2006/05/peer" xmlns:xs="h
ttp://www.w3.org/2001/XMLSchema">
  <xs:import namespace="http://www.w3.org/2005/08/addressing" />
  <xs:import namespace="http://schemas.datacontract.org/2004/07/System
.Net" />
  <xs:element name="Update">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ClientId" type="xs:string" />
        <xs:element name="MeshId" type="xs:string" />
        <xs:element name="NodeAddress" type="PeerNodeAddress" />
        <xs:element name="RegistrationId" type="GUID" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Element Name	Description
ClientId	String identifying the resolver client. This MUST be unique and SHOULD be created upon the creation of the resolver client. The ClientId of a particular resolver client SHOULD be valid for the lifetime of that client.
MeshId	Mesh name.
NodeAddress	PeerNodeAddress structure containing the updated endpoint information to be written at the registration record indicated by **RegistrationId**. See [MC-PRCH] section 2.2.1.2.
RegistrationId	GUID identifying the registration record to be updated. Created by the resolver service as a result of processing a Register message.

2.2.2.5 Unregister Message

The Unregister message is sent by a resolver client to remove its registration record from the service.

```
<wsdl:operation name="Unregister">
  <soap12:operation soapAction="http://schemas.microsoft.com/net/2006/
05/peer/resolver/Unregister" style="document" />
- <wsdl:input name="UnregisterInfo">
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_Unreg
ister_Input_policy" />
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_Unreg
ister_output_policy" />
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>

<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/net/2006/05/peer" a
ttributeFormDefault="unqualified" elementFormDefault="qualified" targe
tNamespace="http://schemas.microsoft.com/net/2006/05/peer" xmlns:xs="h
```

```
ttp://www.w3.org/2001/XMLSchema">
  <xs:element name="Unregister">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="MeshId" type="xs:string" />
        <xs:element name="RegistrationId" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Element Name	Description
MeshId	Mesh name
RegistrationId	Unique string identifying a registration record to be deleted.

2.2.2.6 GetServiceInfo Message

The GetServiceInfo message is sent by the resolver client during the initialization of a PeerChannel client. It is used to determine whether or not the PeerChannel client will use referrals to discover new neighbors.

```
<wsdl:operation name="GetServiceInfo">
  <soap12:operation soapAction="http://schemas.microsoft.com/net/2006/
05/peer/resolver/GetServiceSettings" style="document" />
- <wsdl:input>
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_GetSe
rviceInfo_Input_policy" />
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="ServiceSettingsResponseInfo">
  <wsp:PolicyReference URI="#WSHttpBinding_IPeerResolverContract_GetSe
rviceInfo_output_policy" />
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/net/2006/05/peer" a
ttributeFormDefault="unqualified" elementFormDefault="qualified" targe
tNamespace="http://schemas.microsoft.com/net/2006/05/peer" xmlns:xs="h
ttp://www.w3.org/2001/XMLSchema">
  <xs:element name="GetServiceInfo">
    <xs:complexType>
      <xs:sequence>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2.2.3 Resolver Service Messages

2.2.3.1 RegisterResponse Message

The RegisterResponse message is sent by the resolver service in response to a Register message. It is only sent to the originator of the Register message, and indicates the result of an attempt to create a new registration record.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/net/2006/05/peer" a
ttributeFormDefault="unqualified" elementFormDefault="qualified" targe
tNamespace="http://schemas.microsoft.com/net/2006/05/peer" xmlns:xs="h
ttp://www.w3.org/2001/XMLSchema">
  <xs:element name="Update">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RegistrationId" type="xs:string" />
        <xs:element name="RegistrationLifetime" type="xs:duration" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Element Name	Description
RegistrationId	Unique string identifying the newly-created registration record. Created by the resolver service as a result of a registration request.
RegistrationLifetime	The duration for which the resolver service guarantees that the newly created registration record will not be deleted. Formatted as a xs:duration [XMLSCHEMA].

2.2.3.2 ResolveResponse Message

The ResolveResponse message is sent by the resolver service in response to a Resolve message. It contains endpoint information corresponding to the details of the Resolve request.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/net/2006/05/peer" a
ttributeFormDefault="unqualified" elementFormDefault="qualified" targe
tNamespace="http://schemas.microsoft.com/net/2006/05/peer" xmlns:xs="h
ttp://www.w3.org/2001/XMLSchema">
<xs:element name="ResolveResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Addresses">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="PeerNodeAddress" type="PeerNodeAddress"
/>
            .
            . (one or more PeerNodeAddress elements)
            .
          </xs:sequence>
        </xs:complexType>
```

```
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element Name	Description
Addresses	A list of PeerNodeAddress structures that were associated with the **MeshId** element specified in the Resolve message. There is no limit to the number of PeerNodeAddress elements that can be contained here.

2.2.3.3 RefreshResponse Message

The RefreshResponse message is sent by the resolver service in response to a Refresh message. It is sent to the originator of the Refresh message and indicates the result of an attempt to extend the expiration time of the resolver client's registration record.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/net/2006/05/peer" a
ttributeFormDefault="unqualified" elementFormDefault="qualified" targe
tNamespace="http://schemas.microsoft.com/net/2006/05/peer" xmlns:xs="h
ttp://www.w3.org/2001/XMLSchema">
  <xs:element name="RefreshResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RegistrationLifetime" type="xs:duration" />
        <xs:element name="Result" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Element Name	Description
RegistrationLifetime	The duration the resolver service guarantees that the refreshed registration record will not be deleted. Formatted as an xs:duration data type [XMLSCHEMA].
Result	String indicating the result of the refresh operation. MUST be one of two values: "Success" if the registration was successful "RegistrationNotFound" if the **RegistrationId** element given by the client could not be found by the service.

2.2.3.4 ServiceSettings Message

The ServiceSettings message is sent by the resolver service in response to a GetServiceInfo message. The message contains a Boolean value indicating whether or not referrals are to be used by PeerChannel clients. For more information on PeerChannel and the use of referrals, see [MC-PRCH] section 3.1.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/net/2006/05/peer" a
```

```
ttributeFormDefault="unqualified" elementFormDefault="qualified" targe
tNamespace="http://schemas.microsoft.com/net/2006/05/peer" xmlns:xs="h
ttp://www.w3.org/2001/XMLSchema">
<xs:element name="ServiceSettings">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ControlMeshShape" type="xs:boolean" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element Name	Description
ControlMeshShape	Boolean value indicating how the client should deal with referrals. For more information on the use of referrals in PeerChannel, see [MC-PRCH].

3 Protocol Details

All interactions between the resolver client and resolver service MUST be initiated by the client, and all messages sent by the resolver client (except for Unregister) MUST receive a corresponding response message from the resolver service.

3.1 Resolver Client Details

3.1.1 Abstract Data Model

The resolver client MUST store the following information:

- **Resolver service configuration details**: Implementation-specific. Used to establish a connection with the resolver service. This MUST include the location of the service, the protocol used to establish connection, and any security configuration, if applicable.
- **Mesh Name(s)**: All messages to and from the resolver service perform operations within the context of a mesh name.

The resolver client MUST also store the following for each registration it makes with the resolver service:

- **RegistrationID**: A GUID that identifies the specific registration record stored at the resolver service, and is used to refer to that record for the Update, Unregister, and Refresh operations. The **RegistrationID** element is contained in the RegisterResponse message returned by the resolver service as a result of a successful **Register** operation and is generated by the resolver service.
- **RegistrationLifetime**: Associated with a particular registration record as indicated by its **RegistrationID**. Indicates the amount of time that the registration record will remain at the resolver service before it is eligible to be deleted.

3.1.2 Timers

3.1.2.1 Client Refresh Timer

The client refresh timer is only needed when the resolver client has registered endpoint information with the resolver service and does not wish that information to be deleted.

This timer is used to determine when to call Refresh. The client MUST ensure that the **Refresh** operation is called before the time indicated by the **RegistrationLifetime** element has elapsed. This value is received as part of the RegisterResponse message in response to the initial Register request.

Note that the exact interval used for the timer MAY<2> vary to take into account the network latency between the resolver client and server.

3.1.2.2 Message Response Timer

This timer measures the length of time that an operation requiring a response will wait to receive an answer.

3.1.3 Initialization

The higher-layer application or protocol configuration specifies the following:

- Resolver service location.
- Security protocol to use in order to secure the connection (if applicable).
- MessageTimeOut duration.

3.1.4 Higher-Layer Triggered Events

All higher-layer triggered events follow the following processing procedure:

1. The resolver client MUST be initialized before processing the event.
2. The resolver client MUST establish a connection with the resolver service using the parameters for location, protocol, and security settings set by initialization before any higher-layer events can be processed.
3. The resolver client sends the message relevant to the event.
4. The resolver client starts a message response timer with the duration set to MessageTimeOut.

3.1.4.1 Register

The Register message stores endpoint information at the resolver service. The resolver client MUST follow the procedure outlined in section 3.1.4 and MUST send a Register message to the resolver service.

3.1.4.2 Update

The Update message modifies an existing registration record stored at the resolver service. The resolver client MUST follow the procedure outlined in section 3.1.4 and MUST send an Update message to the resolver service.

3.1.4.3 Resolve

The Resolve operation retrieves registration information corresponding to a specific mesh name. The resolver client MUST follow the procedure outlined in section 3.1.4 and MUST send a Resolve message to the resolver service.

3.1.4.4 Unregister

The Unregister message removes a specific registration record from the resolver service. The resolver client MUST follow the procedure outlined in section 3.1.4, with the exception that it MUST NOT start a message response timer. It MUST send an Unregister message to the resolver service, and MAY<3> close the connection to the resolver service after successfully sending the Unregister message.

3.1.4.5 Refresh

The Refresh message extends the **registration lifetime** of a specific registration. The resolver client MUST follow the procedure outlined in section 3.1.4 and MUST send a Refresh message to the resolver service.

3.1.4.6 GetServiceInfo

The GetServiceInfo operation is used to query the referral policy of a PeerChannel mesh. The resolver client MUST follow the procedure outlined in section 3.1.4 and MUST send a GetServiceInfo message to the resolver service.

3.1.5 Message Processing Events and Sequencing Rules

After an appropriate response message is received (for instance, RegisterResponse for a Register request, and so on), the resolver client:

- MUST clear the message response timer.
- SHOULD close the connection with the resolver service.

3.1.5.1 RegisterResponse Message

If the resolver client is not waiting for a response to a Register or Response message, it MUST ignore the RegisterResponse message.

Upon receiving the RegisterResponse message, the resolver client MUST store the **RegistrationId** value locally. Additionally, the resolver client SHOULD start a client refresh timer with the time interval indicated in the **RegistrationLifetime** value in the RegisterResponse message.

3.1.5.2 ResolveResponse Message

If the resolver client is not waiting for a response to a Resolve message, it MUST ignore this message.

Upon receiving this message, the client MUST deliver the list of **PeerNodeAddress** elements to the higher-layer protocol or application.

3.1.5.3 RefreshResponse Message

If the resolver client is not waiting for a response to a Refresh message, it MUST ignore this message.

The client MUST take the following action depending on the value of the RefreshResult enumeration in the **Result** element in the RefreshResponse message:

- If the RefreshResult value is "Success", the resolver client MUST reset the client refresh timer with the value given by **RegistrationLifetime** in the RefreshResponse message.
- If the RefreshResult value is "RegistrationNotFound", the resolver client MUST ignore the value of the **RegistrationLifetime** field and initiate the **Register** event.

3.1.5.4 ServiceSettingsInfoResponse Message

If the custom resolver client is not waiting for a ServiceSettingsInfoResponse message, it MUST ignore this message.

Otherwise it MUST pass the value of the **ControlMeshShape** field to the higher-level protocol or application.

3.1.6 Timer Events

3.1.6.1 Client Refresh Timer

When the client refresh timer fires, the resolver client MUST initiate a **Refresh** operation.

3.1.6.2 Message Response Timer

If the message response timer fires, it indicates that the operation that started the timer has failed. The client MUST close the connection with the service and signal failure to the higher-layer protocol or application.

3.1.7 Other Local Events

3.1.7.1 Shutting down the Resolver Service

When the resolver client attempts to shut down, it SHOULD send an Unregister request to the resolver service.

3.2 Resolver Service Details

3.2.1 Abstract Data Model

The resolver service MUST store the following information:

- **Registration records**: All registration records that are registered with the service. Each record MUST contain the following fields, as defined in the glossary of this document: **ClientId**, mesh name, RegistrationId, PeerNodeAddress, and Expiration Time. The format of the last field is protocol-independent.
- **MaintenanceInterval**: Defines the periodicity of the maintenance timer. This SHOULD be one minute.
- **DefaultRegistrationLifetime**: Defines the amount of time for which registration records will remain valid before requiring to be refreshed. This SHOULD be set to ten minutes.
- **ReferralPolicy**: Boolean value indicating whether or not referrals are to be used by PeerChannel clients using this resolver service as a discovery mechanism.

3.2.2 Timers

3.2.2.1 Maintenance Timer

The maintenance timer is used to periodically remove stale registration records from the date store of the resolver service.

3.2.3 Initialization

The resolver service MUST start a listener that will accept connections at the specific location and with the specific protocol and security configuration with which resolver clients have been configured.

3.2.4 Higher-Layer Triggered Events

There are no higher-layer triggered events for the resolver service. All operations are confined to the processing of messages (section 3.2.5) or the maintenance timer (section 3.2.2.1).

3.2.5 Message Processing Events and Sequencing Rules

The resolver service MUST send all response messages to the originator of the corresponding request message.

If any request message is incomplete or incorrectly formatted, the resolver service MUST abort the connection with the resolver client immediately.

3.2.5.1 Register

The service MUST create a new registration record using the PeerNodeAddress structure contained in the Register message, and MUST assign the new record a unique **RegistrationId** element. The service then MUST send a RegisterResponse message to the client with the newly-created **RegistrationId** and a **RegistrationLifetime** element with the value of "DefaultRegistrationLifetime".

3.2.5.2 Update

The service MUST attempt to locate the registration record indicated by the **MeshId** and **RegistrationId** fields in the Update message. If the record cannot be found, the resolver service SHOULD create a new record and send a RegisterResponse message, as outlined in section 3.2.5.1. The **RegistrationId** of the new record MAY<4> be different from the value given in the Update message. If the registration record can be found, the resolver service MUST replace the PeerNodeAddress in that record with the PeerNodeAddress provided in the Update message. The service then MUST send a RegisterResponse message back to the resolver client with the **RegistrationId** of the registration record that was updated and **RegistrationLifetime** set to "DefaultRegistrationLifetime".

3.2.5.3 Resolve

The resolver service MUST attempt to locate registration records associated with the value of the **MeshId** element in the Resolve message. The resolver service MAY<5> use the **RegistrationId** or **ClientId** to determine which specific registration records to return. Once these records have been chosen by the implementation-specific mechanism, the resolver service MUST create a ResolveResponse message with the PeerNodeAddress structures of these records and send it to the requesting resolver client.

3.2.5.4 Unregister

The resolver service SHOULD attempt to locate a registration record, as given by the value of the **RegistrationId** element in the Unregister message. If the record is found, it MUST be removed or otherwise marked as expired.

3.2.5.5 Refresh

The resolver service MUST attempt to locate the registration record indicated by the value of the **RegistrationId** element in the Refresh request. If the record is found, the resolver service MUST extend the registration record's expiration time by "DefaultRegistrationLifetime". The resolver service then MUST send a RefreshResponse message to the resolver client with the value of "Success" in the **Result** element and "DefaultRegistrationLifetime" in the **RegistrationLifetime**

element. If the attempt to locate the record is unsuccessful, the resolver service MUST send a RefreshResponse message to the requesting resolver client with the value of "RegistrationNotFound" in the **Result** element and leave the **RegistrationLifetime** field blank.

3.2.5.6 GetServiceInfo

The resolver service MUST return to the requesting resolver client a ServiceSettings message with the local value of "ReferralPolicy" entered for the **ControlMeshShape** element.

3.2.6 Timer Events

3.2.6.1 Maintenance Timer

When the maintenance timer fires, the resolver service MUST examine its data store of registration records. All records whose expiration time precedes the current system time MUST be removed or marked as invalid.

3.2.7 Other Local Events

There are no other local events for the resolver service.

4 Protocol Examples

4.1 Peer Discovery

The primary use of the Peer Channel Custom Resolver Protocol is to allow client applications to discover and connect to other clients. The following example illustrates how the use of this protocol allows two nodes to find and connect to each other.

Terms Used In This Example:

Node "A", Node "B": Two separate applications/nodes running a custom resolver client that implements this protocol.

PNA_A, PNA_B: PeerNodeAddresses corresponding to nodes "A", node "B", and the resolver service respectively.

"ExampleMesh": The mesh name used in this protocol example.

"S" –: A machine running a custom resolver service that implements this protocol.

Discovery Process:

A: Node A Registers with Resolver Service

1. "A" establishes a connection with "S".
2. "A" creates a Register request (contents: PNA_A, "ExampleMesh") and sends this message to "S".
3. "S" receives the Register request and adds PNA_A to its data store under the name "ExampleMesh" with a **RegistrationId** of 100.
4. "S" sends a RegisterResponse message to "A" with the **RegistrationId** of the newly-created registration record (100).
5. "A" receives the RegisterResponse message and stores the **RegistrationId** locally for future use.

B: Node A Resolves for Other Nodes

1. "A" sends a Resolve request to "S" (contents: "ExampleMesh", 5 for **MaxAddresses**).
2. "S" receives the Resolve request and searches its data store for registration records under the mesh name "ExampleMesh". It finds one record (with **RegistrationId** of 100), which contains PNA_A.
3. "S" sends a ResolveResponse message to "A" containing one PeerNodeAddress (PNA_A).
4. "A" receives the ResolveResponse message, and extracts the single PeerNodeAddress PNA_A. Since this record corresponds to its own address, "A" does nothing else at this time. Node "A" may choose to send an additional Resolve request after some time has passed to check for newly-registered nodes.

C: Node B Resolves for Other Nodes

1. "B" sends a Resolve request to "S" (contents: "ExampleMesh", 5 for **MaxAddress**)
2. "S" receives "B"'s Resolver request, searches data store for registration records under mesh name "ExampleMesh", and finds one record (number 100), which contains PNA_A.

3. "S" sends a ResolveResponse message to "B" containing PNA_A.

4. "B" receives the ResolveResponse message, extracts PNA_A from message body.

5. "B" uses PNA_A to set up direct connection between "A" and "B".

D: Node B Registers with the Resolver Service

(Same as section A, substituting node "B" for node "A" and PNA_B for PNA_A).

Note that the order of Register and Resolve is completely arbitrary; it makes no difference whether or not a node registers before or after resolving provided that the node ensures that it does not try to connect to itself. On one hand, it may be advisable to always Register first to avoid having nodes receive their own records from Resolve requests. On the other hand, performing Register first ensures that the node is discoverable by other nodes as soon as possible.

In the example above, if node "B" had registered before resolving, it would have received both PNA_A and PNA_B in the ResolveResponse message.

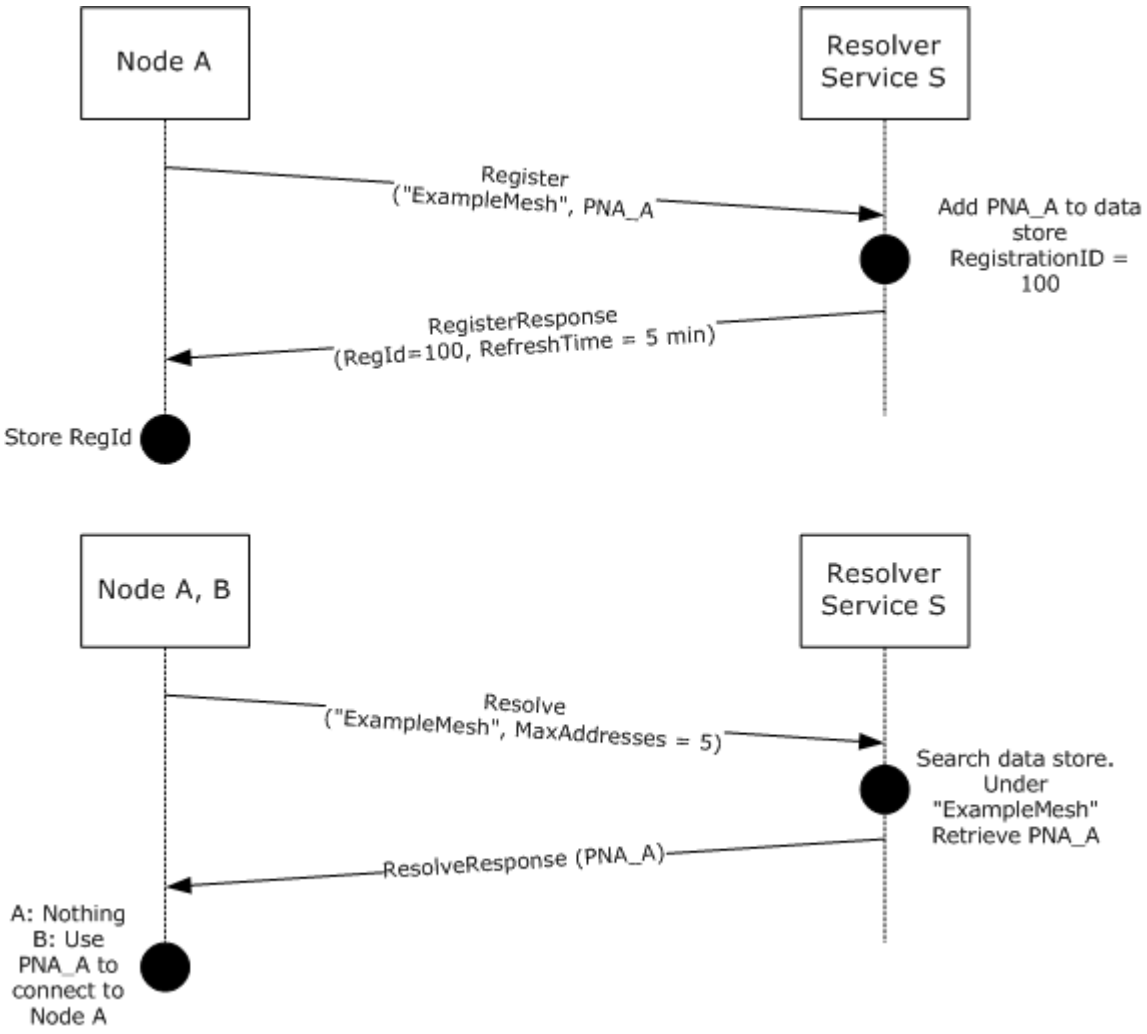

Figure 1: Register and resolve sequences

5 Security

5.1 Security Considerations for Implementers

Although the Peer Channel Custom Resolver Protocol does not include any specific security information, it is advised that the connection between the custom resolver client and the custom resolver service be secured. This is important if the mesh name and IP addresses of clients need to remain private. Additionally, a malicious attacker could use information sent over the wire to corrupt the data store of the resolver service (for example, by prematurely unregistering registration records from the service by sniffing RegistrationIds off the wire).

5.2 Index of Security Parameters

There are no security parameters associated with this protocol.

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows XP
- Windows Server 2003
- Windows Vista

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 2.1: The Windows implementation of the Peer Channel Custom Resolver uses TCP connections.

<2> Section 3.1.2.1: The Windows implementation does not take into account client-server latency when setting the refresh timer.

<3> Section 3.1.4.4: The Windows implementation of the Peer Channel Custom Resolver Protocol closes the connection between the resolver client and resolver service after each response message is received by the client. A new connection is established every time the resolver client wishes to send a request to the service.

<4> Section 3.2.5.2: If an Update message is received by the Windows implementation of the custom resolver service and the RegistrationId cannot be found in the service's data store, a new registration record is created with a unique **RegistrationId**.

<5> Section 3.2.5.3: The Windows implementation of the Peer Channel Custom Resolver service is intended to distribute connections evenly among participating resolver clients. As a result, if the number of registration records under a particular mesh name exceeds the value of the **MaxAddresses** element as contained in the Resolve message, the resolver service chooses a random subset of those registration records to return to the resolver client. The **RegistrationId** and **ClientId** elements in the Resolve message are not used by the Windows implementation of the resolver service.

7 Index

T

V

W