[MS-NRLS]:
.NET Remoting:
Lifetime Services Extension

This protocol documentation is intended for use in conjunction with publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

A protocol specification does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them.

Revision Summary

Date	Revision History	Revision Class	Comments
07/20/2007	0.1	Major	MCPP Milestone 5 Initial Availability
09/28/2007	1.0	Major	Updated and revised the technical content.
10/23/2007	1.0.1	Editorial	Revised and edited the technical content.
11/30/2007	1.0.2	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
01/25/2008	1.1	Minor	Updated the technical content.

Table of Contents

1 Introduction

This document specifies the .NET Remoting: Lifetime Services Extension, a Microsoft proprietary protocol. This protocol adds lifetime and remote **activation** capabilities to the .NET Remoting Core Protocol (specified in [MS-NRTP]). This protocol builds on the [MS-NRTP] specification, and readers must be familiar with its terms and concepts.

1.1 Glossary

The following terms are defined in [MS-NRTP]:

Array
Class
Data Value
Exception
Library
Marshaled Server Object (MSO)
Null Object
Primitive Type
Proxy
Remote Field
Remote Method
Remoting Type
Server Activated Object (SAO)
Server Object
Server Object Reference
Server Object Table
Server Object URI
Server Type
Singleton SAO
System Library

The following terms are specific to this document:

Activation: The process of creating a **Server Object**.

Client Activated Object (CAO): A **Marshaled Server Object (MSO)** that requires an explicit **activation** message to create the **Server Object**.

Lease Object: A Lease Object is a type of **MSO**. Every **Singleton SAO** and **MSO** has a Lease Object associated which contains methods that control the lifetime of the **Server Object**. It must be noted that although a Lease Object is an **MSO** it does not have a Lease Object of its own. The lifetime of the Lease Object is bound by the lifetime of the associated **Server Object**.

Sponsor: An **MSO** that is implemented by clients to participate in the renewal process of a **Server Object's** lifetime.

Time-To-Live (TTL): The time duration for which a **Server Object** is available.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as specified 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.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-NRTP] Microsoft Corporation, ".NET Remoting: Core Protocol Specification", September 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

1.2.2 Informative References

[MSDN-.NETFrameWrk] Microsoft Corporation, ".NET Framework", http://msdn2.microsoft.com/en-us/netframework/default.aspx

[MSDN-RemotingLifetime] Microsoft Corporation, "Managing the Lifetime of Remote .NET Objects with Leasing and Sponsorship", http://msdn.microsoft.com/msdnmag/issues/03/12/LeaseManager/default.aspx

1.3 Protocol Overview (Synopsis)

The .NET Remoting Core Protocol (specified in [MS-NRTP]) defines mechanisms for the creation of **Server Objects** and the invocation of **Remote Methods** on those Server Objects.

This protocol adds a mechanism that allows clients to explicitly create Server Objects and another mechanism that allows clients and servers to control the lifetime of Server Objects.

Additional overview information for the .NET Remoting: Lifetime Services Extension is available in the following sections:

- Section 1.3.1 - Activating a server from a client
- Section 1.3.2 - Managing the connection lifetime between a client and a server
- Section 1.3.3 - Managing the **sponsors** (clients) associated with a server

Much of the basic information and terminology used in this document is also common to the .NET Remoting Core Protocol. For more information, see [MS-NRTP] Appendix B.

1.3.1 Client Activation

This protocol introduces a new type of Server Object called a **Client Activated Object (CAO)**. A CAO can be remotely activated by a client by invoking the **Activate** Remote Method on a well-known **Server Activated Object (SAO)**, passing the **Server Type**. The implementation of the SAO creates a new instance of the Server Type, registers it in the **Server Object Table**, and sends a **Server Object Reference** to the instance back to the client. The client receives the Server Object Reference and can use it to create a **Proxy** to invoke methods on the CAO.

An example of a client activating an instance and invoking a Remote Method, **increment()**, is shown below.

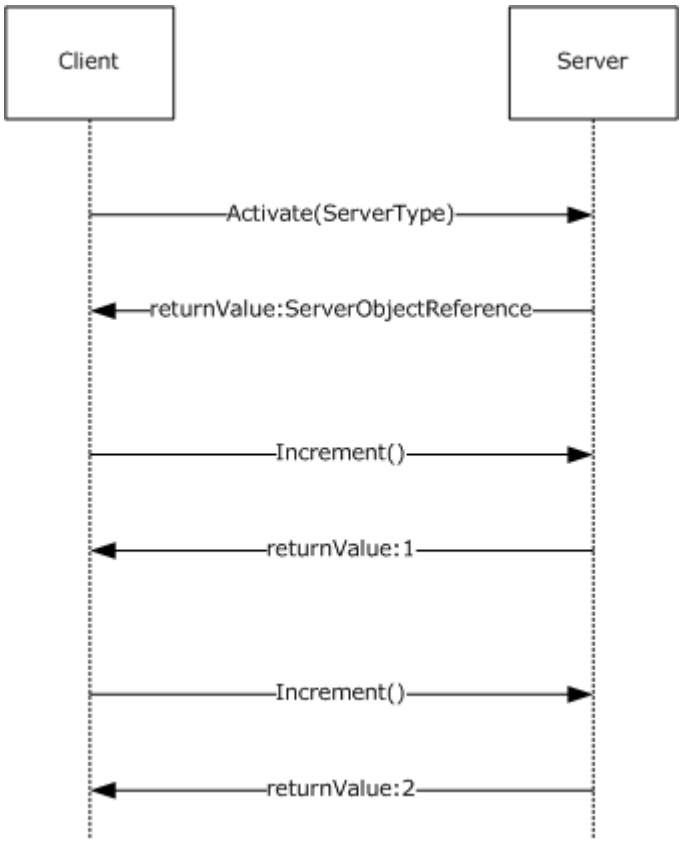

Figure 1: Client activating a server object

1.3.2 Lifetime Management

This protocol specifies a lease-based model for lifetime management of **Marshaled Server Objects (MSOs)** and the **Singleton SAO**.

A **Lease Object** is associated with each Server Object. Each Lease Object has an initial **Time-To-Live (TTL)** for the Server Object. For every Remote Method invocation on the Server Object, the TTL is extended. If no calls are made to the Server Object for the duration of the TTL, the Server Object is considered for removal from the Server Object Table.

A client can explicitly control the Server Object's lifetime through Remote Method invocations on the Server Object's Lease Object. The client gets a Server Object Reference to the Lease Object for a Server Object by calling the Server Object's **GetLifetimeServiceRequest** Remote Method. The client can then invoke the **Renew** Remote Method on the Lease Object to extend the TTL by a desired amount.

1.3.3 Sponsor

A Lease Object for a given Server Object maintains a list of Sponsors that are called when the TTL of the Server Object expires. Each Sponsor can specify whether the Server Object's TTL must be extended, and specify the duration of the extension. If there are no associated Sponsors or if none

of the associated Sponsors extend the lifetime of the Server Object, then the Server Object is removed from the Server Object Table, making it unavailable to clients.

An example of a client managing the lifetime of a Server Object is shown in the following examples.

The client invokes a Remote Method on the Server Object:

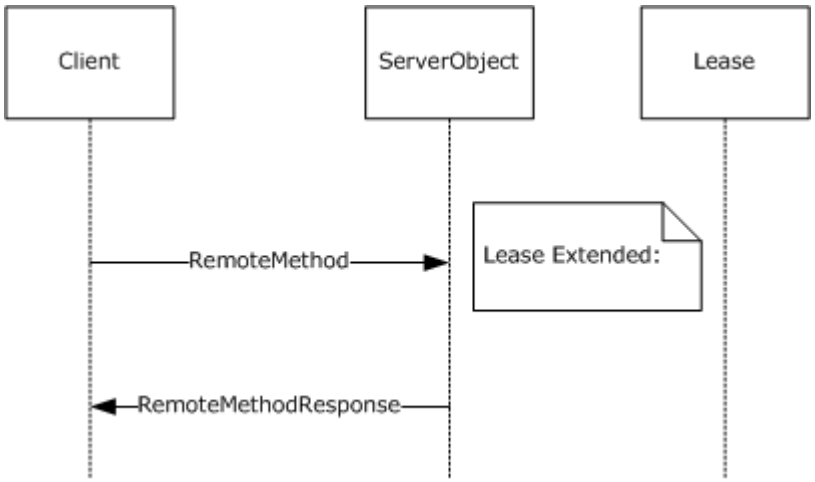

Figure 2: Invoking a Remote Method on the Server Object

The client uses the Lease Object to extend the lease time:

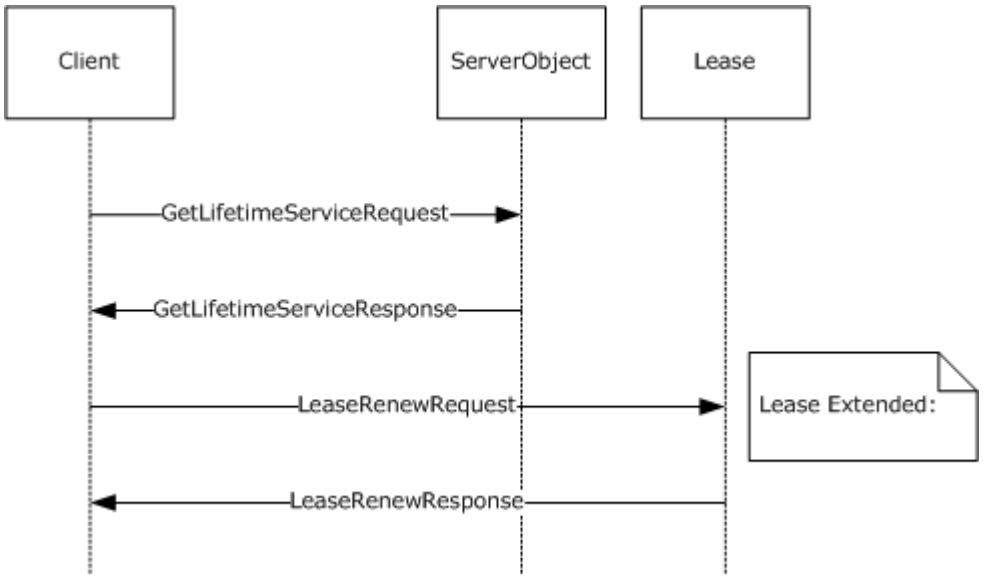

Figure 3: Extending lease time

The client registers a Sponsor that is invoked when the Lease Object's TTL expires:

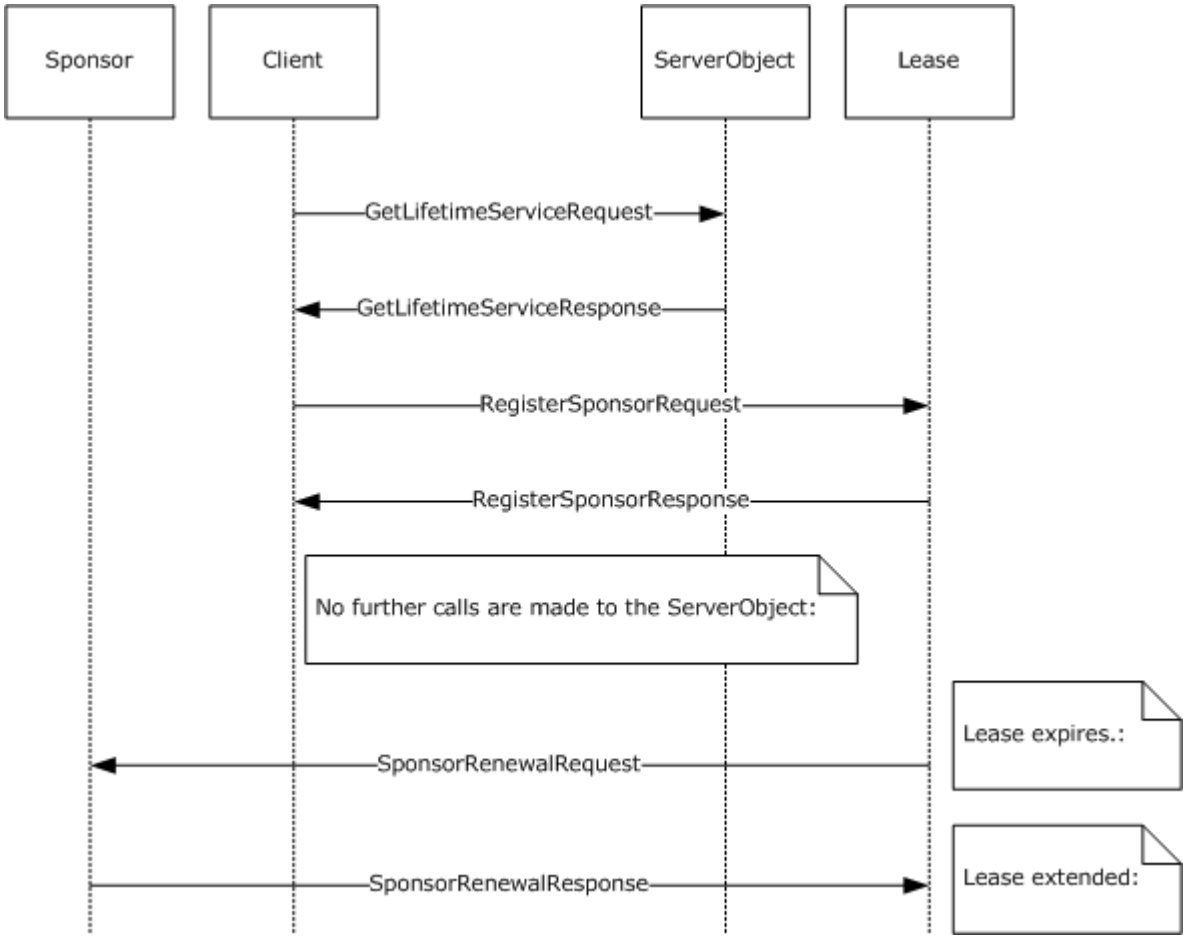

Figure 4: Registering a Sponsor

For more information on how leases and Sponsors are exposed in Windows, see [MSDN-RemotingLifetime].

1.3.4 Notational Conventions

Remoting Type definitions in this document use the notation defined in [MS-NRTP]Appendix B.

1.4 Relationship to Other Protocols

This protocol extends the .NET Remoting Core Protocol, adding new methods for activation and lifetime management.

The protocol layering of the related protocols is shown below.

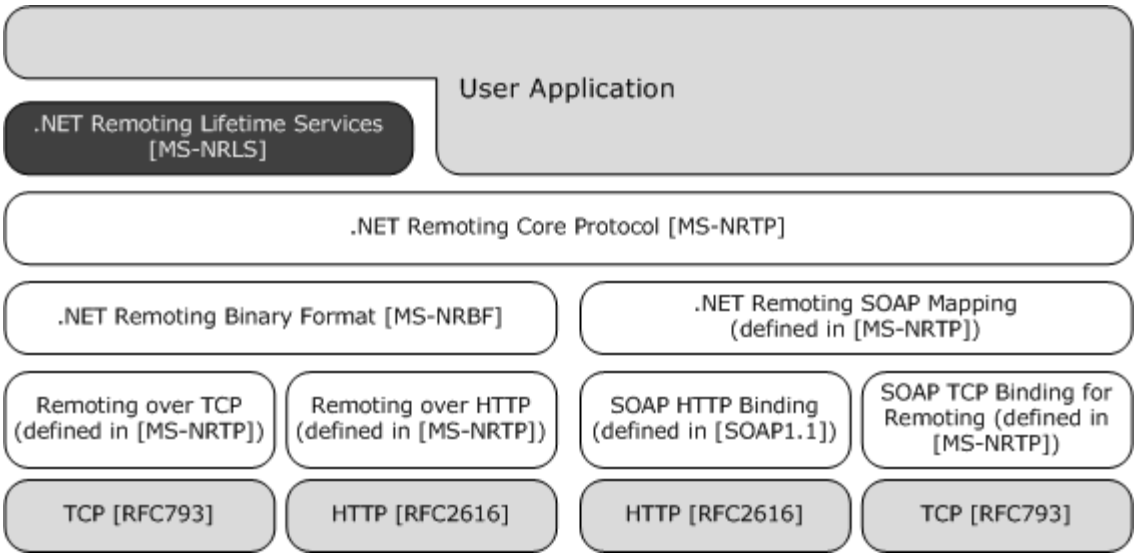

Figure 5: NRLS protocol stack

1.5 Prerequisites/Preconditions

This protocol layers on top of the .NET Remoting Core Protocol and, as a result, has the prerequisites specified in [MS-NRTP]. In addition, for a Client Activated Object (CAO), the client application must be configured with enough information about the Server Type to construct the activation message.

1.6 Applicability Statement

The protocol described here is applicable to users of the .NET Remoting Core Protocol in environments that require distributed activation and lifetime management of Server Objects.

CAOs require a server to maintain references to each client object created, which might not scale to large numbers of clients.

The Sponsor mechanism requires that references be maintained from each server to all registered Sponsors. In addition, the server must individually contact each client with a Sponsor, which does not scale for large numbers of clients holding Sponsors.

1.7 Versioning and Capability Negotiation

This protocol has no versioning or capability negotiation.

1.8 Vendor-Extensible Fields

This protocol has no vendor-extensible fields.

1.9 Standards Assignments

There are no standards assignments made by this protocol.

2 Messages

The following sections specify message relationships to the .NET Remoting: Core Protocol, as well as common .NET Remoting: Lifetime Services Extension Remoting Types.

2.1 Transport

This protocol can be bound to any transport supported by the .NET Remoting Core Protocol, as specified in [MS-NRTP] section 2.1.

2.2 Common Data Types

2.2.1 ArrayList

ArrayList is a **Class**. The **Library** name of the Class is "mscorlib". It represents a collection of **Data Values**. The capacity of the collection is increased dynamically as required.

```
namespace System.Collections
{
  class ArrayList
  {
    System.Object[]   _items;
    Int32             _size;
    Int32             _version;
  }
}
```

_items: An **Array** that holds Data Values. The size of the Array MUST be greater than or equal to the value of the **_size** field.

_size: An Int32 value that indicates the number of items present in the ArrayList.

_version: An Int32 value that is unused by this protocol. It MAY contain any value and the value MUST be ignored.<1>

Note The Array is resized as new items are added to the collection. To accommodate adding items in a performant way, the size of the Array MAY be more than the number of items in the collection. If an element of the _items Array has an index greater than or equal to the value of the **_size** field, it is not considered part of the ArrayList. The element MAY contain any value and the value MUST be ignored.<2>

2.2.2 ConstructionCall

ConstructionCall is a Class. The Library name of the Class is "mscorlib". It is used to activate a Server Object.

```
namespace System.Runtime.Remoting.Messaging
{
  class ConstructionCall
  {
    String                          __Uri;
    String                          __MethodName;
    System.Type[]                   __MethodSignature;
```

```
        String                          __TypeName;
        System.Object[]                 __Args;
        System.Object                   __CallContext;
        System.Type                     __ActivationType;
        System.Object                   __Activator;
        String                          __ActivationTypeName;
        System.Collections.ArrayList    __ContextProperties;
        System.Object[]                 __CallSiteActivationAttributes;
    }
}
```

__Uri: A string value that is unused by this protocol. It MAY contain any value and the value MUST be ignored.<3>

__MethodName: A string value that specifies the name of the Remote Method. Its value MUST be ".ctor".

__MethodSignature: An Array of type System.Type. Each item in the Array contains information about the Remoting Type of the arguments that are needed to create an instance of the Server Object. System.Type is defined in [MS-NRTP] section 2.2.2.11.

__TypeName: A string value that contains the name of the Server Type to activate.

__Args: An Array of Objects that contains the parameters required to create an instance of the Server Object.

__CallContext: A **Null Object**. This field is reserved in this protocol. The value of this field MUST be NullObject.

__ActivationType: A Null Object, or an instance assignable to System.Type that contains information about the Server Type that is being activated. Its value SHOULD be a Null Object.

__Activator: An Object field that is unused in the protocol. This field MAY contain any value and the value MUST be ignored.<4>

__ActivationTypeName: A String value that contains the name of the Server Type. This field MUST have the same value as the field **__TypeName**.

__ContextProperties: An ArrayList that contains additional values required for the activation of the Server Type. The interpretation of the values is higher-layer–defined. If there are no properties, this value MUST be an empty ArrayList (that is, an ArrayList with a value of 0 for the **_size** field).<5>

__CallSiteActivationAttributes: A Null Object, or an Array of any Data Values. The interpretation of the values is higher-layer–defined. If there are no values, then this value MUST be a Null Object.<6>

2.2.3 ContextLevelActivator

ContextLevelActivator is a Class. The Library name of the Class is "mscorlib". It is used in the __Activator field of a ConstructionCall instance.

```
namespace System.Runtime.Remoting.Activation
{
```

```
  class ContextLevelActivator
  {
    System.Runtime.Remoting.Activation.ConstructionLevelActivator
m_NextActivator;
  }
}
```

m_NextActivator: An instance of ContextLevelActivator.

2.2.4 ConstructionLevelActivator

ConstructionLevelActivator is a Class. The Library name of the Class is "mscorlib". It is used in the **m_NextActivator** field of a ContextLevelActivator instance.

```
namespace System.Runtime.Remoting.Activation
{
  class ConstructionLevelActivator
  {
  }
}
```

This Class has no members.

2.2.5 ConstructionResponse

ConstructionResponse is a Class. The Library name of the Class is "mscorlib". It is used to activate a Server Object.

```
namespace System.Runtime.Remoting.Messaging
{
  class ConstructionResponse
  {
    String                    __Uri;
    String                    __MethodName;
    String                    __TypeName;
    System.Object                    __Return;
    System.Object[]                  __OutArgs;
    System.Object                    __CallContext;
  }
}
```

__Uri: The field MAY contain any value and the value MUST be ignored.<7>

__MethodName: A string value that specifies the name of the Remote Method. Its value MUST be ".ctor".

__TypeName: A string value that contains the name of the Server Type that was activated.

__Return: This field contains the ObjRef (as specified in [MS-NRTP] section 2.2.2.1) for the activated Server Object.

__OutArgs: The value of this field MUST be an Array of System.Object. The length of the Array MUST be 0.

__CallContext: The value of this field MUST be NullObject.

2.2.6 LeaseState

The LeaseState enumeration provides state information about a Lease Object. The size of this enumeration is an Int32.

```
namespace System.Runtime.Remoting.Lifetime
{
  enum LeaseState : Int32
  {
     Null     = 0,
     Initial  = 1,
     Active   = 2,
     Renewing = 3,
     Expired  = 4
  }
}
```

Null: The Lease Object is in an error state.

Initial: This is the initial state when the Lease Object is created.

Active: The Lease Object is actively maintaining the lifetime of Server Object.

Renewing: The TTL has expired and is in the process of renewing.

Expired: The Lease Object has expired.

3 Protocol Details

The client side of this protocol is simply a pass-through. That is, no additional timers or other state is required on the client side of this protocol. Calls made by the higher-layer protocol or application are passed directly to the transport, and the results returned by the transport are passed directly back to the higher-layer protocol or application.

This protocol extends the server role defined in the .NET Remoting Core Protocol Server Details ([MS-NRTP] section 3.2) in the following ways:

- The server implementation MUST register a Server Activated Object (SAO) that implements the **IActivator (section 3.1)** interface. The **Server Object URI** of the SAO MUST be "RemoteActivationService.rem".
- Each Server Object participating in the lifetime management MUST implement **MarshalByRefObject (section 3.2)**.
- The **ILease** Abstract Data Model (section 3.3.1) extends the .NET Remoting Core Protocol Server Abstract Data Model ([MS-NRTP] section 3.2.1) in the following ways:
  - Associates a Lease Object with each Server Object in the Server Object Table.
  - Updates the TTL of the Lease Object on each invocation of an application-defined Remote Method.

This protocol extends the client role defined in the .NET Remoting Core Protocol Client Details ([MS-NRTP] section 3.3) in the following way:

- The client MAY implement the **ISponsor (section 3.4)** interface to participate in the lifetime management of the Server Object.<8>

3.1 IActivator

3.1.1 Abstract Data Model

There is no Abstract Data Model for this interface.

3.1.2 Timers

There are no timers associated with this interface.

3.1.3 Initialization

A Singleton SAO MUST be registered as specified in [MS-NRTP] section 3.2.4.1. The Server Object URI MUST be "RemoteActivationService.rem".

3.1.4 Message Processing Events and Sequencing Rules

This interface includes the following method:

Method	Description
Activate	Activates the specified Server Object

3.1.4.1 Activate

Activateactivates a Server Object. The parameter specifies the Server Type of the Server Object.

```
System.Runtime.Remoting.Messaging.ConstructionResponse
Activate(
  System.Runtime.Remoting.Messaging.ConstructionCall callMessage);
```

callMessage: An instance of ConstructionCall that contains information that is required to activate the Server Object.

Return Values: An instance of ConstructionResponse that contains the ObjRef (as specified in [MS-NRTP] section 2.2.2.1) to the activated Server Object.

Exceptions:

- If the Server Object cannot be activated, a RemotingException (specified in [MS-NRTP] section 2.2.2.9) is thrown.
- If the *callMessage* parameter does not fulfill the constraints specified in ConstructionCall, then a RemotingException MUST be constructed as specified in [MS-NRTP] section 3.2.5.5.2. The **exception** MUST be sent back to the client.

  An implementation MAY have implementation-specific rules to determine whether the **__ActivationType** is valid.<9>

A Server Object is constructed as follows:

- The implementation MAY use any information from the ConstructionCall message to determine and then construct an appropriate Server Type.
- If the implementation is unable to construct the Server Object, then a RemotingException MUST be constructed as specified in [MS-NRTP] section 3.2.5.5.2. The exception MUST be sent back to the client.<10>

An implementation MUST construct a ConstructionResponse as specified in section 2.2.5, with the following additional constraints:

- The **__Return** field of the ConstructionResponse MUST be set to the newly created Server Object.
- The **__MethodName** MUST match the **__MethodName** field of the incoming ConstructionCall instance.
- The **__TypeName** MUST match the **__TypeName** field of the incoming ConstructionCall instance.

The ConstructionResponse instance MUST be sent back as the return value of the method.

3.1.5 Timer Events

There are no timer events associated with this interface.

3.1.6 Other Local Events

There are no other local events.

3.2 MarshalByRefObject

3.2.1 Abstract Data Model

This protocol extends the [MS-NRTP] Abstract Data Model (as specified in [MS-NRTP] sections 3.1.1 and 3.2.1) to associate a Lease Object with every active Server Object.

3.2.2 Timers

There are no timers associated with this interface.

3.2.3 Initialization

A Lease Object and Lease Object Data MUST be created and associated with a Singleton SAO or MSO during initialization.

3.2.4 Message Processing Events and Sequencing Rules

This interface includes the following methods.

Method	Description
GetLifetimeService	Returns an ObjRef to the Lease Object associated with the target Server Object.
FieldGetter	Returns the value of the specified field.
FieldSetter	Sets the value of the specified field to the specified value.

3.2.4.1 GetLifetimeService

GetLifetimeService retrieves a reference to the Lease Object associated with the target Server Object.

```
System.Runtime.Remoting.Lifetime.ILease GetLifetimeService();
```

Return Values: An ObjRef (as specified in [MS-NRTP] section 2.2.2.1) to a Lease Object from which a Proxy to the Lease Object can be created.

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

On the first call to **GetLifetimeService**, the implementation MUST do the following:

- Set the CurrentLeaseState of the Lease Object associated with the target Server Object to Active.
- Set the CurrentLeaseState of the associated Lease Data to Active.

A Server Object Reference for the new Lease Object MUST be created as specified in [MS-NRTP] section 3.2.5.3.3. The ObjRef (as specified in [MS-NRTP] section 2.2.2.1) that represents the Server Object Reference MUST be returned as the return value for the method.

3.2.4.2 FieldGetter

FieldGetter returns the value of the specified **Remote Field**.

```
void FieldGetter(String typeName, String fieldName, ref
System.Object val);
```

typeName: A string value that specifies the name of the Server Interface containing the Remote Field. The Server Interface MUST be the Server Type of the Server Object.

fieldName: A string value that specifies the name of the Remote Field whose value is to be retrieved. The Remote Field MUST be defined in the Server Interface specified by the **typeName** field.

val: The value of the Remote Field. This is a ref argument. Its value on input MUST be ignored. An implementation MUST set the argument to the value of the Remote Field.

Exceptions: If the Remote Field specified in the **fieldName** argument is not defined in the Server Interface specified by the **typeName** argument, then a RemotingException (as specified in [MS-NRTP] section 2.2.2.7) MUST be sent back.

3.2.4.3 FieldSetter

FieldSetter sets the value of the specified Remote Field to the specified value.

```
void FieldSetter(String typeName, String fieldName,
System.Object val);
```

typeName: A string value that specifies the name of the Type containing the Remote Field. The Type MUST be the Type or base Type of the Server Object.

fieldName: A string value that specifies the name of the Remote Field whose value is to be set. The Remote Field MUST be the defined in the Type specified by the **typeName** field.

val: The value of the field. An implementation MUST set the value of the Remote Field to the value of this argument.

Exceptions: If the Remote Field specified in the **fieldName** argument is not defined in the Type specified by the **typeName** argument, then a RemotingException (as specified in [MS-NRTP] section 2.2.2.7) MUST be sent back.

3.2.5 Timer Events

There are no timer events associated with this interface.

3.2.6 Other Local Events

There are no other local events.

3.3 ILease

3.3.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

Sponsor Info

Sponsor Info contains the following information about a Sponsor:

- **Proxy:** A Proxy to a Sponsor that was registered by a client.
- **RenewalTime:** The TimeSpan value that was passed when the Sponsor was registered.

Lease Data

Lease Data extends the Server Object Table defined in [MS-NRTP] section 3.2.1.

Lease Data is associated with the Lease Object of a Server Object. Lease Data contains the following values:

- **InitialLeaseTime:** The initial TTL of a Server Object when it is marshaled.
- **RenewOnCallTime:** The duration by which to extend the TTL when a method is called in the associated Server Object.
- **SponsorCallTimeout:** The duration to wait for a Sponsor to respond.
- **CurrentLeaseState:** A LeaseState value that indicates the current state of the Lease Object.
- **SponsorList:** A list of Sponsor Info. The list is sorted in decreasing order of the Sponsor Infos' **RenewalTime** field values.

3.3.2 Timers

Lease TTL Timer: tracks the TTL of a Server Object. Each Lease Object is associated with a Lease TTL Timer that fires when its TTL expires.

3.3.3 Initialization

An implementation MUST set the initial values of InitialLeaseTime, RenewOnCallTime and SponsorCallTimeout to a nonzero positive value. The initial value of the CurrentLeaseState MUST be *Initial*.<11>

3.3.4 Message Processing Events and Sequencing Rules

This interface includes the following methods.

Method	Description
Renew	Increases the TTL by the specified amount.

Method	Description
Register **Register(Overload)**	Registers the specified Sponsor in a Lease Object's SponsorList. Two forms of this method exist.
Unregister	Unregisters a Sponsor from the Lease Object's SponsorTable.
get_InitialLeaseTime	Returns the Lease Object's InitialLeaseTime.
set_InitialLeaseTime	Updates the Lease Object's InitialLeaseTime with the specified amount.
get_RenewOnCallTime	Returns the Lease Object's RenewOnCallTime.
set_RenewOnCallTime	Updates the Lease Object's RenewOnCallTime.
get_SponsorShipTimeout	Returns the Lease Object's SponsorshipTimeout.
set_SponsorShipTimeout	Updates the Lease Object's SponsorshipTimeout.
get_CurrentLeaseTime	Returns the time when the Lease Object expires.
get_CurrentLeaseState	Returns the Lease Object's current state.

Lease Data's CurrentLeaseState determines whether a method can be called. Calling the above operations takes the Lease Object through various LeaseStates. The state machine that captures the LeaseState transitions is specified in the following diagram.

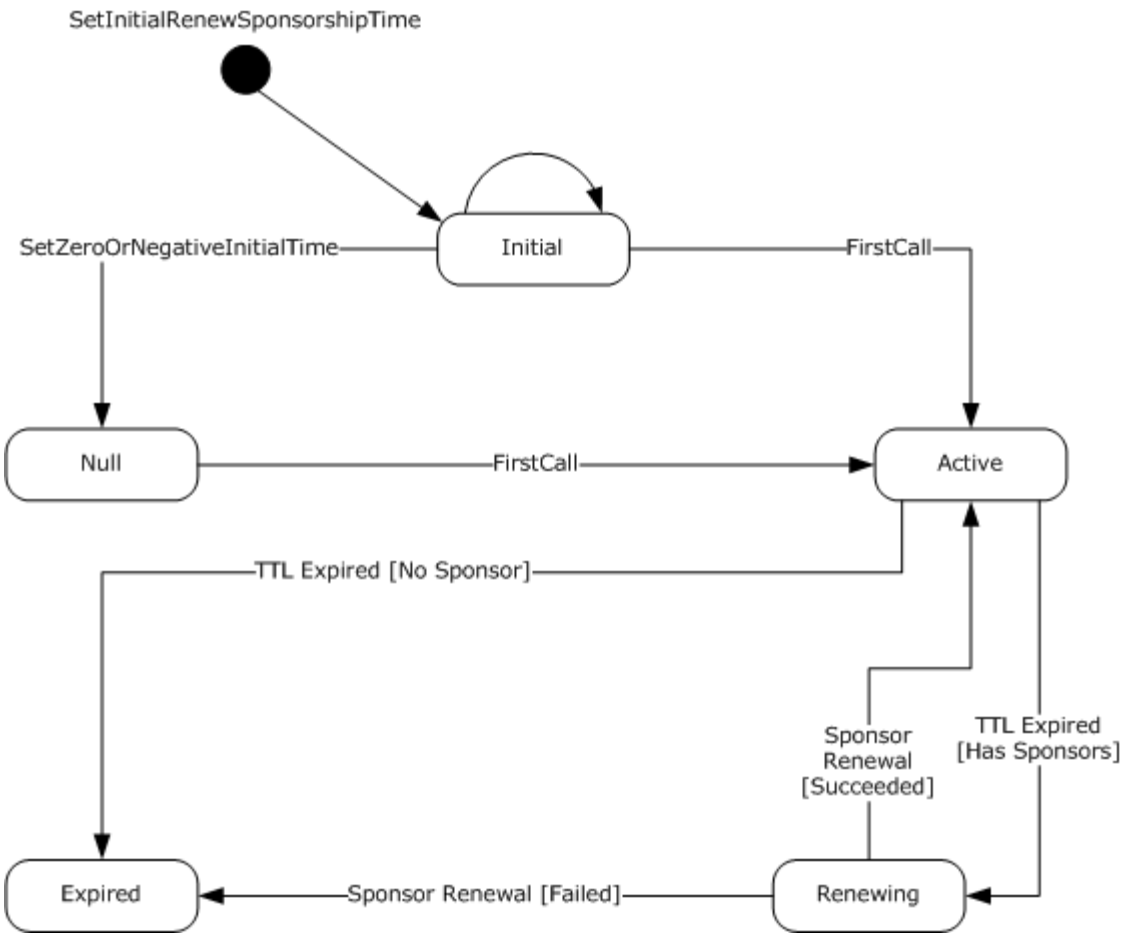

Figure 6: Lease state machine

3.3.4.1 Renew

Renew extends the TTL of a Server Object.

```
TimeSpan Renew(
     TimeSpan renewalTime
);
```

renewalTime: A TimeSpan value that specifies the required TTL for the Server Object.

Return Value: A TimeSpan value that specifies the new TTL for the Server Object.

Exceptions: No exception other than the exceptions that are specified in [MS-NRTP] section 3.

If the CurrentLeaseState value of the associated Lease Data is "Expired", then the implementation MUST not modify the TTL; instead the implementation MUST return a TimeSpan of 0.

If the CurrentLeaseState is valid, the new TTL MUST be computed as follows:

- If the remaining TTL is less than the renewalTime, the renewal time MUST be the new TTL and the Lease Timer MUST be updated to fire when the new TTL expires.
- Otherwise, the remaining TTL MUST be left as is.

The implementation MUST return the new TTL.

3.3.4.2 Register

Register registers a Sponsor with the Lease Object associated with the Server Object.

```
void Register(
    System.Runtime.Remoting.Lifetime.ISponsor sponsor
);
```

sponsor: A Proxy to a Server Object that implements the ISponsor interface.

Return Values: There are no return values for this method.

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

The implementation of the method MUST create a new Sponsor Info with Sponsor instance referenced by the argument *sponsor*, and a TimeSpan value of 0. The Sponsor Info MUST be added to the end of the SponsorList of the associated Lease Data.

3.3.4.3 Register(Overload)

Register(Overload) registers a Sponsor with the Lease Object associated with the Server Object. This implementation of the method includes a second parameter, "renewalTime".

```
void Register (
    System.Runtime.Remoting.Lifetime.ISponsor sponsor,
    TimeSpan renewalTime
);
```

sponsor: A Proxy to a Server Object that implements the ISponsor interface.

renewalTime: A TimeSpan value that specifies the required TTL for the Server Object.

Return Values: There are no return values for this method

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

The implementation of the method MUST add the Sponsor to the end of the SponsorList of the associated Lease Data.

The implementation of the method MUST:

- Create a new Sponsor Info with the Sponsor instance referenced by the argument *sponsor*, and the TimeSpan value of the argument renewalTime. The Sponsor Info MUST be inserted in the

SponsorList of the associated Lease Data such that the items in the list continue to be sorted in decreasing order of the Sponsor Info's renewalTime.

- Extend the CurrentLeaseTime of the associated Lease Data with the renewal TimeSpan as specified in the **Renew** method.

3.3.4.4 Unregister

Unregister removes the specified Sponsor from the Sponsor List.

```
void Unregister(
    System.Runtime.Remoting.Lifetime.ISponsor sponsor
);
```

sponsor: An ObjRef (as specified in [MS-NRTP] section 2.2.2.1) to a registered Sponsor that needs to be unregistered.

Return Values: There are no return values for this method.

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

Two Sponsor Proxies are considered identical if their Server Object URIs match. The Server Object URI can be looked up for a given Proxy in the Proxy Table, as specified in [MS-NRTP] section 3.3.1.

If the SponsorList of the associated Lease Data contains a Sponsor Info with Sponsor Proxy identical to the one referenced by the *sponsor* argument, the implementation MUST remove the Sponsor Info from the SponsorList.

3.3.4.5 get_InitialLeaseTime

get_InitialLeaseTime returns the Lease Object's InitialLeaseTime.

```
TimeSpan get_InitialLeaseTime();
```

Return Values: A TimeSpan value that is the InitialLeaseTime in the associated Lease Data.

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

The method has no arguments. The implementation of the method MUST return the InitialLeaseTime of the associated Lease Data.

3.3.4.6 set_InitialLeaseTime

set_InitialLeaseTime updates the Lease Object's InitialLeaseTime with a specified value.

```
void set_InitialLeaseTime(TimeSpan value);
```

value: A TimeSpan value that has to be set as the InitialLeaseTime in the associated Lease Data.

Return Values: There are no return values for this method.

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

If the associated Lease Data's CurrentLeaseState is not "Initial", then a RemotingException MUST be constructed as specified in [MS-NRTP] section 3.2.5.5.2, and the exception MUST be sent back to the client; otherwise, the InitialLeaseTime MUST be set to the value of the argument *value*.

If the new TimeSpan value is negative, the CurrentLeaseState MUST be set to Null state.

3.3.4.7 get_RenewOnCallTime

get_RenewOnCallTime returns the Lease Object's RenewOnCallTime.

```
TimeSpan get_RenewOnCallTime();
```

Return Values: A TimeSpan value that is the RenewOnCallTime in the associated Lease Data.

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

The method has no arguments. The implementation of the method MUST return the RenewOnCallTime of the associated Lease Data.

3.3.4.8 set_RenewOnCallTime

set_RenewOnCallTime updates the Lease Object's RenewOnCallTime.

```
void set_RenewOnCallTime(TimeSpan value);
```

value: A TimeSpan value that has to be set as the RenewOnCallTime in the associated Lease Data.

Return Values: There are no return values for this method.

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

If the associated Lease Data's CurrentLeaseState is not "Initial", then a RemotingException MUST be constructed as specified in the [MS-NRTP] section 3.2.5.5.2, and the exception MUST be sent back to the client; otherwise, the argument *value* MUST be set as the new value of RenewOnCallTime.

3.3.4.9 get_SponsorshipTimeout

get_SponsorshipTimeout returns the Lease Object's SponsorshipTimeout.

```
TimeSpan get_SponsorshipTimeout();
```

Return Values: A TimeSpan value that is the SponsorshipTimeout in the associated Lease Data.

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

The method has no arguments. The implementation of the method MUST return the SponsorshipTimeout of the associated Lease Data.

3.3.4.10 set_SponsorshipTimeout

set_SponsorshipTimeout updates the Lease Object's SponsorshipTimeout.

```
void set_SponsorshipTimeout(System.Timespan value);
```

value: A TimeSpan value that has to be set as the SponsorshipTimeout in the associated Lease Data.

Return Values: There are no return values for this method

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

If the associated Lease Data's CurrentLeaseState is not "Initial", then a RemotingException MUST be constructed as specified in [MS-NRTP] section 3.2.5.5.2 and the exception MUST be sent back to the client; otherwise, the argument *value* MUST be set as the new value of SponsorshipTimeout.

3.3.4.11 get_CurrentLeaseTime

get_CurrentLeaseTime returns the expiration time of the Lease Object.

```
TimeSpan get_CurrentLeaseTime();
```

Return Values: A TimeSpan value that is the TTL of the associated Server Object.

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

The method has no arguments. The implementation of the method MUST return the time at which the TTL of the associated Server Object ends.

3.3.4.12 get_CurrentLeaseState

get_CurrentLeaseState returns the current LeaseState value of the Lease Object.

```
System.Runtime.Remoting.Lifetime.LeaseState get_CurrentLeaseState();
```

Return Values: A LeaseState value that is the CurrentLeaseState of the associated LeaseData.

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

The method has no arguments. The implementation of the method MUST return the CurrentLeaseState of the associated Lease Data.

3.3.5 Timer Events

3.3.5.1 Lease TTL Timer

When the Lease TTL Timer is fired, an implementation of the protocol MUST evaluate lease renewal as specified below.

If there are no Sponsor Info instances in the Lease Data's SponsorList, the implementation MUST set the CurrentLeaseState of the associated Lease Data to "Expired", and MUST unmarshal the Server Object, as specified in [MS-NRTP] section 3.2.4.3.

If there are Sponsor Infos in the Lease Data's SponsorList, then the **Renewal** method of the first Sponsor Info's Proxy MUST be called. If the method returns successfully and the return value is a time duration greater than 0, the implementation MUST do the following:

- Extend the TTL of the Server Object by that TimeSpan.
- Reset the timer to fire after the new TTL.
- Set the Sponsor Info's RenewalTime field to the time duration that was returned.
- Reposition the Sponsor Info with the new RenewalTime in the SponsorList, such that the list is sorted in decreasing order of RenewalTime.

If any of the following conditions occur, the renewal call is considered unsuccessful, and the implementation must remove the Sponsor Info from the SponsorList.

- The **Renewal** method did not return within the duration specified in SponsorCallTimeout.
- The **Renewal** method threw an exception.
- The **Renewal** method returned a TimeSpan of 0.

If the renewal call was unsuccessful, the implementation MUST repeat the renewal process with the next Sponsor Info in the SponsorList.

If there are no Sponsor Infos left in the SponsorList, the implementation MUST do the following:

- Set the CurrentLeaseState of the Lease Data to "Expired".
- Unmarshal the associated Server Object as specified in [MS-NRTP] section 3.2.4.3.
- Unmarshal the Lease Object as specified in [MS-NRTP] section 3.2.4.3.

3.3.6 Other Local Events

3.3.6.1 Binding to Server Object

This protocol augments [MS-NRTP] section 3.2.5.1 to specify additional processing for lifetime management. When a request is bound to a Server Object TTL, its Lease Object MUST be extended by RenewOnCallTime as specified in the ILease Renew method.

3.4 ISponsor

3.4.1 Abstract Data Model

There is no data model for this interface.

3.4.2 Timers

There are no timers beyond those provided by the underlying transport layers.

3.4.3 Initialization

There is no initialization required by the implementation of this interface.

3.4.4 Message Processing Events and Sequencing Rules

This interface includes the following method.

Method	Description
Renewal	Extends the TTL of the associated Server Object

3.4.4.1 Renewal

Renewal extends the TTL of the associated Server Object.

```
TimeSpan Renewal();
```

Return Values: There are no return values for this method.

Exceptions: The .NET Remoting: Lifetime Services Extension has no defined exceptions for this method; however, exceptions specified in [MS-NRTP] section 3 are valid.

This method has no arguments. An implementation of this method MUST return a TimeSpan which is the new TTL. The implementation MUST return a valid TimeSpan value greater than or equal to 0. The meaning of the return value is specified in the following table.

Value	Meaning
0	Lease need not be renewed, and this Sponsor can be dropped from the Lease SponsorList.
>0	Lease needs to be renewed, and this Sponsor needs to be kept in the Lease SponsorList.

3.4.5 Timer Events

There are no timer events.

3.4.6 Other Local Events

There are no other events.

4 Protocol Examples

The following sections provide common scenarios to illustrate the function of the .NET Remoting: Lifetime Services Extension.

4.1 CAO Activation Request/Response message.

This sample shows the messages involved when the client sends an Activation request for a CAO where the transport is TCP and the format is Binary.

The client is requesting the server to activate an instance of ServerType "DOJRemotingMetadata.MyServer". The Server Type is defined in the Library "DOJRemotingMetadata". This is done by calling the **Activate** method in System.Runtime.Remoting.Activation.IActivator Server Type. The Server Object URI is "RemoteServiceActivation.rem". The server is hosted on machine "maheshdev2".

The server is configured to support TCP on port 8080 and the messages are expected to be encoded in the .NET Remoting Binary Format, as specified in [MS-NRBF].

The sequence diagram for an Activation process is shown in the following figure.

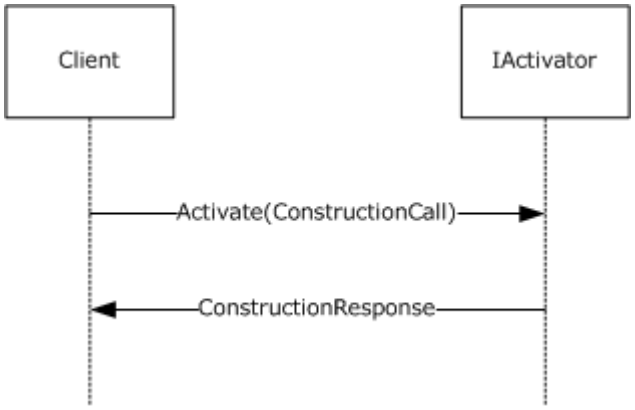

Figure 7: Activation process

4.1.1 Activation Request Message

The client passes a ConstructionCall (section 2.2.2) instance as an argument to the **Activate** method of the **IActivator** interface.

The sample message structure of the Activation request is shown below.

```
ProtocolIdentifier: 0x54454E2E
MajorVersion: 1 (0x1)
MinorVersion: 0 (0x0)
Operation: Request (0x00)
Content Length
   Content Distribution: Content Length (0x00)
   Content Length: 1013 (0x03F5)
Header 1:
   RequestUriHeader
      HeaderToken: RequestUri (0x04)
      DataType:  CountedString (0x01)
```

```
      StringEncoding: UTF8 (0x01)
      UriValue: tcp://maheshdev2:8080/RemoteActivationService.rem
Header 2:
   ContentTypeHeader:
      HeaderToken: ContentType (0x06)
      DataType: CountedString (0x01)
      ContentTypeValue: application/octet-stream
Header 3:
   EndHeader:
      HeaderToken: EndOfHeaders(0x00)

Binary Serialization Format
   SerializationHeaderRecord:
      BinaryHeaderEnum: SerializedStreamHeader (0x00)
      TopId: 1 (0x1)
      HeaderId: -1 (0xFFFFFFFF)
      MajorVersion: 1 (0x1)
      MinorVersion: 0 (0x0)
   BinaryMethodCall:
      BinaryHeaderEnum: BinaryMethodCall (0x15)
      MessageEnum: 00000012
         NoArgs:                 (.............................0)
         ArgsInline:             (.............................0.)
         ArgsIsArray:            (............................1..)
         ArgsInArray:            (...........................0...)
         NoContext:              (..........................1....)
         ContextInline:          (.........................0.....)
         ContextInArray:         (........................0......)
         MethodSignatureInArray: (.......................0.......)
         PropertyInArray:        (......................0........)
         NoReturnValue:          (.....................0.........)
         ReturnValueVoid:        (....................0..........)
         ReturnValueInline:      (...................0...........)
         ReturnValueInArray:     (..................0............)
         ExceptionInArray:       (.................0.............)
         Reserved:               (0000000000000000..............)
      MethodName:
         PrimitiveTypeEnum: String (0x12)
         Data: Activate
      TypeName:
         PrimitiveTypeEnum: String (0x12)
         Data: System.Runtime.Remoting.Activation.IActivator,
               mscorlib, Version=2.0.0.0, Culture=neutral,
               PublicKeyToken=b77a5c561934e089
      ArgsCount: 0 (0x0)

   CallArray:
      ArraySingleObject:
         ObjectId: 1 (0x1)
         Length: 1 (0x1)
      MemberReference:
         IdRef: 2
         SystemClassWithMembersAndTypes:
         BinaryHeaderEnum: SystemClassWithMembersAndTypes (0x04)
         ObjectId: 2
         Name: System.Runtime.Remoting.Messaging.ConstructionCall
         NumMembers: 11 (0x0B)
            MemberNames:
```

```
            Data: __Uri
         MemberNames:
            Data: __MethodName
         MemberNames:
            Data: __MethodSignature
         MemberNames:
            Data: __TypeName
         MemberNames:
            Data: __Args
         MemberNames:
            Data: __CallContext
         MemberNames:
            Data: __CallSiteActivationAttributes
         MemberNames:
            Data: __ActivationType
         MemberNames:
            Data: __ContextProperties
         MemberNames:
            Data: __Activator
         MemberNames:
            Data __ActivationTypeName
   BinaryTypeEnumA:
      Object  (0x02)
      String  (0x01)
      SystemClass (0x03)
      String  (0x01)
      ObjectArray (0x05)
      Object (0x02)
      Object (0x02)
      Object (0x02)
      SystemClass (0x03)
      SystemClass (0x03)
      String  (0x01)
   AdditionalTypeInformationArray:
      SystemClass:
         Length: 13 (0x0D)
         Data: System.Type[]
      SystemClass:
         Length: 28 (0x1C)
         Data: System.Collections.ArrayList
      SystemClass:
         Length: 56 (0x38)
         Data: System.Runtime.Remoting.Activation.
               ContextLevelActivator
ObjectNull:
   BinaryHeaderEnum: ObjectNull (0x0A)
BinaryObjectString:
   BinaryHeaderEnum: BinaryObjectString (0x06)
   ObjectId: 3 (0x03)
   Length: 5 (0x05)
   Value: .ctor
MemberReference:
   BinaryHeaderEnum: MemberReference (0x09)
   IdRef: 4 (0x04)
BinaryObjectString:
   BinaryHeaderEnum: BinaryObjectString (0x06)
   ObjectId: 5 (0x05)
   Length: 111 (0x06F)
```

```
      Value: DOJRemotingMetadata.MyServer, DOJRemotingMetadata,
             Version=1.0.2616.21414, Culture=neutral,
             PublicKeyToken=null
   MemberReference:
      BinaryHeaderEnum: MemberReference (0x09)
      IdRef: 6 (0x06)
   ObjectNull:
      BinaryHeaderEnum: ObjectNull (0x0A)
   ObjectNull:
      BinaryHeaderEnum: ObjectNull (0x0A)
   ObjectNull:
      BinaryHeaderEnum: ObjectNull (0x0A)
   MemberReference:
      BinaryHeaderEnum: MemberReference (0x09)
      IdRef: 7 (0x07)
   MemberReference:
      BinaryHeaderEnum: MemberReference (0x09)
      IdRef: 8 (0x08)
   MemberReference:
      BinaryHeaderEnum: MemberReference (0x09)
      IdRef: 5 (0x05)
   BinaryArray:
      BinaryHeaderEnum: BinaryArray (0x07)
      ObjectId: 4 (0x04)
      BinaryArrayTypeEnum: 0 (0x0)
      Rank: 1 (0x1)
      LengthA: 0 (0x0)
      BinaryTypeEnumA:
         ObjectUrt (0x03)
            Length: 11 (0x0B)
            Value: System.Type
   SystemClassWithMembersAndTypes:
      BinaryHeaderEnum: SystemClassWithMembersAndTypes (0x04)
      ObjectId: 7 (0x07)
      Name: System.Collections.ArrayList
      NumMembers: 3 (0x03)
         MemberNames:
            Data: _items
         MemberNames:
            Data: _size
         MemberNames:
            Data: _version
      BinaryTypeEnumA:
         ObjectArray (0x05)
         Primitive  (0x00)
         Primitive  (0x00)
      Additional Type Information:
         Primitive Type: Int32 (0x08)
         Primitive Type: Int32 (0x08)
   Object Information Array:
      MemberReference:
         BinaryHeaderEnum: MemberReference (0x09)
         IdRef: 10 (0x0A)
      MemberPrimitiveUnTyped:
         Value: 0 (0x00)
      MemberPrimitiveUnTyped:
         Value: 0 (0x00)
   SystemClassWithMembersAndTypes:
```

```
      BinaryHeaderEnum: SystemClassWithMembersAndTypes (0x04)
      ObjectId: 8
      Name: System.Runtime.Remoting.Activation.ContextLevelActivator
      NumMembers: 1 (0x01)
         MemberNames:
            Data: m_NextActivator
      BinaryTypeEnumA:
         ObjectUrt (0x03)
      Additional Type Information:
         ObjectUrt:
            Length: 61 (0x3D)
            Data: System.Runtime.Remoting.Activation.
                  ConstructionLevelActivator
   MemberReference:
      BinaryHeaderEnum: MemberReference (0x09)
      IdRef: 11 (0x0B)
   ArraySingleObject:
      BinaryHeaderEnum: ArraySingleObject (0x10)
      ObjectId: 10 (0x0A)
      Length: 0 (0x00)
   BinaryObjectWithMapTyped:
      BinaryHeaderEnum: SystemClassWithMembersAndTypes (0x04)
      ObjectId: 11
      Name: System.Runtime.Remoting.Activation.
            ConstructionLevelActivator
      NumMembers: 0 (0x00)
   MessageEnd:
      BinaryHeaderEnum: MessageEnd (0x0B)
```

4.1.2 Activation Response Message

The server sends back the ObjRef (as specified in [MS-NRTP] section 2.2.2.1) of the activated object as part of the ConstructionResponse instance that is the return value. The ObjRef that is passed contains the following information.

Data type	Description
URI	/8dabf534_bf0d_4429_a333_d2216f111d90/iLImNXo5ioIkQjrVqx+SkAtj_1.rem
TypeInfo	Type information for the object
ChannelInfo	Contains information about two channels: ▪ CrossAppDomainData ▪ ChannelDataStore [tcp://172.30.184.185:8080]

The ObjRef in this sample indicates that the Server Object activated is hosted on the relative address of "8dabf534_bf0d_4429_a333_d2216f111d90/iLImNXo5ioIkQjrVqx+SkAtj_1.rem". CrossAppDomainData is an intra-process channel and can be ignored. The ObjRef can be accessed via a TCP connection to port 8080 on IP address "172.30.184.185".

```
ProtocolIdentifier: 0x54454E2E
MajorVersion: 1 (0x1)
```

```
MinorVersion: 0 (0x0)
Operation: Response (0x02)
Content Length
   Content Distribution : Content Length (0x00)
   Content Length: 1269 (0x04F5)
Header 1:
   EndHeader:
      HeaderToken: EndOfHeaders(0x00)
Binary Serialization Format
   SerializationHeaderRecord:
      BinaryHeaderEnum: SerializedStreamHeader (0x00)
      TopId: 1 (0x1)
      HeaderId: -1 (0xFFFFFFFF)
      MajorVersion: 1 (0x1)
      MinorVersion: 0 (0x0)
   BinaryMethodReturn:
      BinaryHeaderEnum: BinaryMethodReturn (0x16)
      MessageEnum: 00001011
         NoArgs:                 (...............................1)
         ArgsInline:             (..............................0.)
         ArgsIsArray:            (.............................0..)
         ArgsInArray:            (............................0...)
         NoContext:              (...........................1....)
         ContextInline:          (..........................0.....)
         ContextInArray:         (.........................0......)
         MethodSignatureInArray: (........................0.......)
         PropertyInArray:        (.......................0........)
         NoReturnValue:          (......................0.........)
         ReturnValueVoid:        (.....................0..........)
         ReturnValueInline:      (....................0...........)
         ReturnValueInArray:     (...................1............)
         ExceptionInArray:       (..................0.............)
         Reserved:               (00000000000000000..............)
  CallArray:
   ArraySingleObject:
      BinaryHeaderEnum: ArraySingleObject (0x10)
      ObjectId: 1 (0x1)
      Length: 1 (0x1)
   MemberReference:
      IdRef: 2
   SystemClassWithMembersAndTypes:
      BinaryHeaderEnum: SystemClassWithMembersAndTypes (0x04)
      ObjectId: 2
      Name: System.Runtime.Remoting.Messaging.ConstructionResponse
      NumMembers: 6 (0x06)
         MemberNames:
            Data: __Uri
         MemberNames:
            Data: __MethodName
         MemberNames:
            Data: __TypeName
         MemberNames:
            Data: __Return
         MemberNames:
            Data: __OutArgs
         MemberNames:
            Data: __CallContext
      BinaryTypeEnumA:
```

```
      Object  (0x02)
      String  (0x01)
      String  (0x01)
      SystemClass (0x03)
      ObjectArray (0x05)
      Object  (0x02)
   AdditionalTypeInformationArray:
      SystemClass:
         Length: 30 (0x1E)
         Data: System.Runtime.Remoting.ObjRef
ObjectNull:
   BinaryHeaderEnum: ObjectNull (0x0A)
BinaryObjectString:
   BinaryHeaderEnum: BinaryObjectString (0x06)
   ObjectId: 3 (0x03)
   Length: 5 (0x05)
   Value: .ctor
BinaryObjectString:
   BinaryHeaderEnum: BinaryObjectString (0x06)
   ObjectId: 4 (0x03)
   Length: 111 (0x6F)
   Value: DOJRemotingMetadata.MyServer, DOJRemotingMetadata,
          Version=1.0.2616.21414, Culture=neutral,
          PublicKeyToken=null
MemberReference:
   BinaryHeaderEnum: MemberReference (0x09)
   IdRef: 5 (0x05)
MemberReference:
   BinaryHeaderEnum: MemberReference (0x09)
   IdRef: 6 (0x06)
ObjectNull:
   BinaryHeaderEnum: ObjectNull (0x0A)
SystemClassWithMembersAndTypes:
   BinaryHeaderEnum: SystemClassWithMembersAndTypes (0x04)
   ObjectId: 5
   Name: System.Runtime.Remoting.ObjRef
   NumMembers: 6 (0x06)
      MemberNames:
         Data: uri
      MemberNames:
         Data: objrefFlags
      MemberNames:
         Data: typeInfo
      MemberNames:
         Data: envoyInfo
      MemberNames:
         Data: channelInfo
      MemberNames:
         Data: fIsMarshalled
   BinaryTypeEnumA:
      String  (0x01)
      PrimitiveTypeEnum  (0x00)
      SystemClass (0x03)
      SystemClass (0x03)
      SystemClass (0x03)
      PrimitiveTypeEnum  (0x00)
   AdditionalTypeInformationArray:
      SystemClass:
```

```
            Length: 32 (0x20)
            Data: System.Runtime.Remoting.TypeInfo
         SystemClass:
            Length: 34 (0x22)
            Data: System.Runtime.Remoting.IEnvoyInfo
         SystemClass:
            Length: 34 (0x22)
            Data: System.Runtime.Remoting.ChannelInfo
   BinaryObjectString:
      BinaryHeaderEnum: BinaryObjectString (0x06)
      ObjectId: 7 (0x07)
      Length: 68 (0x44)
      Value: /8dabf534_bf0d_4429_a333_d2216f111d90/
             iLImNXo5ioIkQjrVqx+SkAtj_1.rem
   MemberPrimitiveUnTyped:
      BinaryTypeEnum: Primitive (0x00)
      Int32Value: 0 (0x00)
   MemberReference:
      BinaryHeaderEnum: MemberReference (0x09)
      IdRef: 8 (0x08)
   ObjectNull:
      BinaryHeaderEnum: ObjectNull (0x0A)
   MemberReference:
      BinaryHeaderEnum: MemberReference (0x09)
      IdRef: 9 (0x09)
   MemberPrimitiveUnTyped:
      Int32Value: 0 (0x00)
   ArraySingleObject:
      BinaryHeaderEnum: ArraySingleObject (0x10)
      ObjectId: 6 (0x06)
      Length: 0 (0x0)
   SystemClassWithMembersAndTypes:
      BinaryHeaderEnum: SystemClassWithMembersAndTypes (0x04)
      ObjectId: 8 (0x08)
      Name: System.Runtime.Remoting.TypeInfo
      NumMembers: 3 (0x03)
         MemberNames:
            Data: serverType
         MemberNames:
            Data: serverHierarchy
         MemberNames:
            Data: interfacesImplemented
      BinaryTypeEnumA:
         String  (0x01)
         StringArray  (0x06)
         StringArray  (0x06)
   BinaryObjectString:
      BinaryHeaderEnum: BinaryObjectString (0x06)
      ObjectId: 10 (0x0A)
      Length: 111 (0x6F)
      Value: DOJRemotingMetadata.MyServer, DOJRemotingMetadata,
             Version=1.0.2616.21414, Culture=neutral,
             PublicKeyToken=null
   BinaryObjectWithMapTyped:
      BinaryHeaderEnum: SystemClassWithMembersAndTypes (0x04)
      ObjectId: 9 (0x09)
      Name: System.Runtime.Remoting.ChannelInfo
      NumMembers: 1 (0x01)
```

```
      MemberNames:
         Data: channelData
   BinaryTypeEnumA:
      ObjectArray  (0x05)
MemberReference:
   BinaryHeaderEnum: MemberReference (0x09)
   IdRef: 11 (0x0B)
ArraySingleObject:
   BinaryHeaderEnum: ArraySingleObject (0x10)
      ObjectId: 11 (0x0B)
      Length: 2 (0x2)
MemberReference:
   BinaryHeaderEnum: MemberReference (0x09)
   IdRef: 12 (0x0C)
MemberReference:
   BinaryHeaderEnum: MemberReference (0x09)
   IdRef: 13 (0x0D)
BinaryObjectWithMapTyped:
   BinaryHeaderEnum: SystemClassWithMembersAndTypes (0x04)
   ObjectId: 12 (0x0C)
   Name: System.Runtime.Remoting.Channels.CrossAppDomainData
   NumMembers: 3 (0x03)
      MemberNames:
         Data: _ContextID
      MemberNames:
         Data: _DomainID
      MemberNames:
         Data: _processGuid
   BinaryTypeEnumA:
      PrimitiveTypeEnum  (0x00)
      PrimitiveTypeEnum  (0x00)
      String  (0x01)
   AdditionalTypeInformationArray:
      PrimitiveTypeEnum: Int32 (0x08)
      PrimitiveTypeEnum: Int32 (0x08)
MemberPrimitiveUnTyped:
   BinaryTypeEnum: Primitive (0x00)
   Int32Value: 1363808 (0x14CF60)
MemberPrimitiveUnTyped:
   BinaryTypeEnum: Primitive (0x00)
   Int32Value: 1 (0x01)
BinaryObjectString:
   BinaryHeaderEnum: BinaryObjectString (0x06)
   ObjectId: 14 (0x0E)
   Length: 111 (0x6F)
   Value: ac118c52_2f96_4034_9af2_e924215f659b
SystemClassWithMembersAndTypes:
   BinaryHeaderEnum: BinaryObjectWithMapTyped (0x04)
   ObjectId: 13 (0x0D)
   Name: System.Runtime.Remoting.Channels.ChannelDataStore
   NumMembers: 2 (0x02)
      MemberNames:
         Data: _channelURIs
      MemberNames:
         Data: _extraData
   BinaryTypeEnumA:
      StringArray  (0x06)
      ObjectUrt (0x03)
```

```
      AdditionalTypeInformationArray:
         ObjectUrt:
            Length: 36 (0x24)
            Data: System.Collections.DictionaryEntry[]
   MemberReference:
      BinaryHeaderEnum: MemberReference (0x09)
      IdRef: 15 (0x0F)
   ObjectNull:
      BinaryHeaderEnum: ObjectNull (0x0A)
   ArraySingleObject:
      BinaryHeaderEnum: ArraySingleObject (0x10)
      ObjectId: 15 (0x0F)
      Length: 1 (0x1)
   BinaryObjectString:
      BinaryHeaderEnum: BinaryObjectString (0x06)
      ObjectId: 16 (0x10)
      Length: 25 (0x19)
      Value: tcp://172.30.184.185:8080
   MessageEnd:
      BinaryHeaderEnum: MessageEnd (0x0B)
```

4.2 Registering a Sponsor for a CAO Object

This sample shows the sequence of steps involved when the client registers a Sponsor to manage the lifetime of the remote CAO Server Object.

1. The client creates a CAO by sending an Activation request to the RemoteActivationService, and receives the Proxy to the Server Object in the Activation response.
2. After the client has the Proxy, it retrieves the Proxy to the CAO's Lease Object by calling the **GetLifetimeService** method.
3. It then registers a Sponsor object by calling the **Register** method.
4. When the TTL of the Server Object expires, the **Renewal** method is called.

The sequence diagram for the above sample is shown in the following figure.

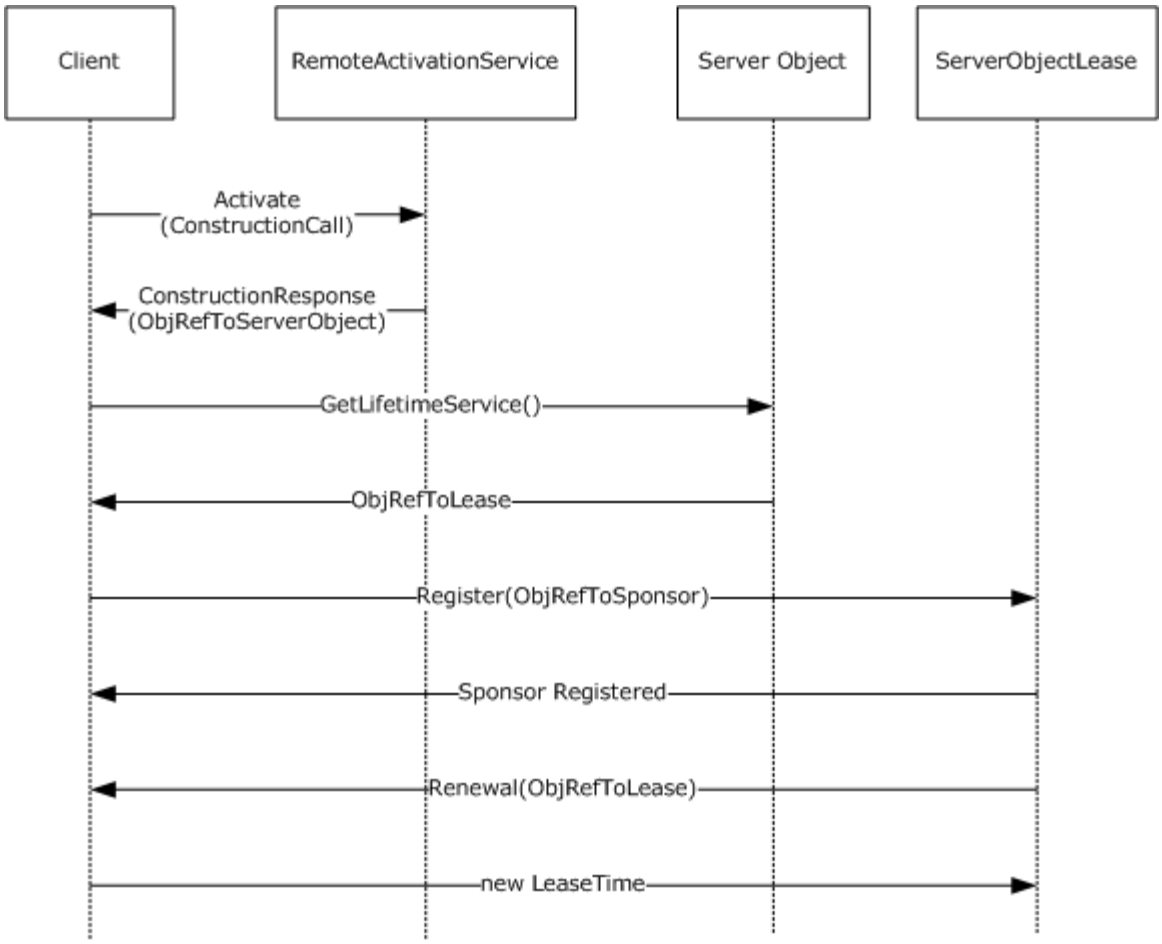

Figure 8: Registering a Sponsor

4.3 Incrementing TTL of a Server Object

The client does not need to register a Sponsor to manage the lifetime of the remote Server Object.

Instead, the client can extend the TTL of the associated Server Object by using the **Renew** method of the Lease Object.

The following diagram illustrates the process of extending a lease by using the **Renew** method.

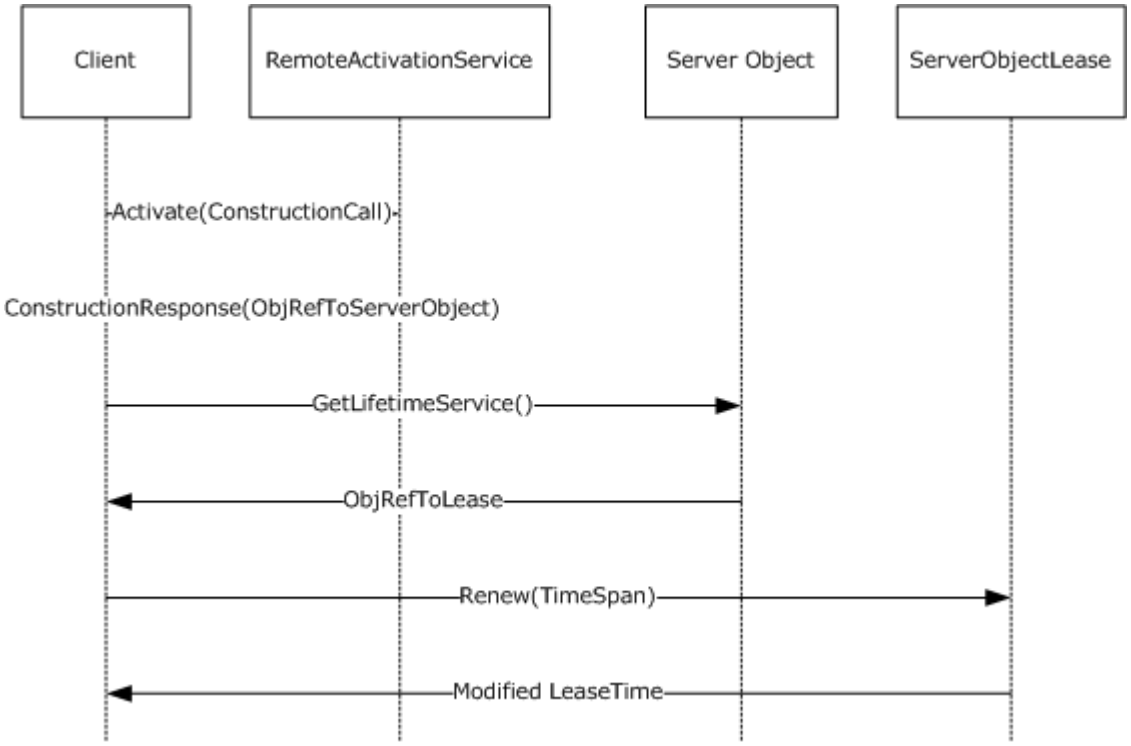

Figure 9: Extending a Lease TTL

5 Security

The following sections specify security considerations for implementers of the .NET Remoting: Lifetime Services Extension.

5.1 Security Considerations for Implementers

This protocol allows a client to request that the server activate a local object by name. This could potentially result in the client being able to run arbitrary code on the server.

Implementers can safeguard against this threat by restricting the set of Remoting Types a client can request to those that are known to be safe—for example, by maintaining a list of allowable Remoting Types that the application can configure.

5.2 Index of Security Parameters

This protocol has no security parameters.

6 Appendix A: Full Definitions

For ease of implementation, the complete definitions of Remoting Types and Server Interfaces are provided below.

The following **Primitive Types** are defined in [MS-NRTP] Appendix B and in the Windows Protocols Master Glossary ([MS-GLOS]):

- Int32
- Object
- String
- TimeSpan

The remainder of this appendix lists the definitions of the elements that comprise the .NET Remoting: Lifetime Services Extension:

```
namespace System.Collections
{
  class ArrayList
  {
    System.Object[]  _items;
    Int32             size;
    Int32             version;
  }
}

namespace System.Runtime.Remoting.Lifetime
{
  enum LeaseState : Int32
  {
    Null     = 0,
    Initial  = 1,
    Active   = 2,
    Renewing = 3,
    Expired  = 4
  }
}

namespace System.Runtime.Remoting.Messaging
{
  class ConstructionCall
  {
    String                          Uri;
    String                        __MethodName;
    System.Type[]                 __MethodSignature;
    String                          TypeName;
    System.Object[]                 Args;
    System.Object                   CallContext;
    System.Type                     ActivationType;
    System.Object                 __Activator;
    String                        __ActivationTypeName;
    System.Collections.ArrayList    ContextProperties;
    System.Object[]                 CallSiteActivationAttributes;
  }

  class ConstructionResponse
  {
    String                          Uri;
    String                        __MethodName;
```

```
    String                          TypeName;
    System.Object                   __Return;
    System.Object[]                 __OutArgs;
    System.Object                   CallContext;
  }
}

namespace System.Runtime.Remoting.Activation
{
  interface IActivator
  {
    System.Runtime.Remoting.Messaging.ConstructionResponse
      Activate(
        System.Runtime.Remoting.Messaging.ConstructionCall callMessage);
  }
}

namespace System
{
  interface MarshalByRefObject
  {
    System.Runtime.Remoting.ILease GetLifetimeService();
  }
}

namespace System.Runtime.Lifetime
{
  interface ILease
  {
    TimeSpan Renew(TimeSpan renewalTime);
    void     Register(System.Runtime.Remoting.ISponsor sponsor);
    void     Register(System.Runtime.Remoting.ISponsor sponsor,
                      TimeSpan renewalTime);
    void     Unregister(System.Runtime.Remoting.ISponsor sponsor);
    TimeSpan get InitialLeaseTime();
    void     set InitialLeaseTime(TimeSpan value);
    TimeSpan get RenewOnCallTime();
    void     set_RenewOnCallTime(TimeSpan value);
    TimeSpan get_SponsorshipTimeout();
    void     set SponsorshipTimeout(System.Timespan value);
    TimeSpan get CurrentLeaseTime();
    System.Runtime.Remoting.LeaseState get CurrentLeaseState();
  }

  interface ISponsor
  {
    TimeSpan Renewal();
  }
}
```

7 Appendix B: Windows Behavior

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

- Windows NT 4.0
- Windows 2000
- 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 Windows does not follow the prescription.

<1> Section 2.2.1: Windows uses this value locally to contain a count of modifications to _items.

<2> Section 2.2.1: Windows writes a Null Object for any elements of the _items Array with an index greater than or equal to the value of the **_size** field.

<3> Section 2.2.2: Windows sets the value of this field to Null Object.

<4> Section 2.2.2: Windows uses this field locally to hold implementation-specific objects. Windows provides an extension mechanism for the higher layer to provide the value of this field. The default value of this field is an instance of ContextLevelActivator (section 2.2.3) Class.

<5> Section 2.2.2: Windows provides an extension mechanism for the higher layer to associate a collection of values with a Server Type that is activated by the client.

<6> Section 2.2.2: Windows provides an extension mechanism for the higher layer to associate a collection of values with a Server Type that is activated by the client.

<7> Section 2.2.5: Windows sets the value of this field to Null Object.

<8> Section 3: In Windows, the higher-layer protocol can provide an implementation of **ISponsor** to participate in the lifetime management of the Server Object.

<9> Section 3.1.4.1: Windows has a list of Remoting Types that can be activated. If the Remoting Type in the ConstructionCall is not found in this list, a RemotingException (as specified in [MS-NRTP] section 2.2.2.8) is sent back.

<10> Section 3.1.4.1: Windows determines the Server Type in the following ways:

- If the **__ActivationType** field is not a Null Object, the value of the **__ActivationType** field is the Server Type.
- If the **__ActivationType** field is a Null Object, the **__ActivationTypeName** field is used to identify the Server Type.

Windows constructs a Server Object from the Server Type using the constructor specified by the **__MethodName** and **__MethodSignature** fields. Windows throws a RemotingException (specified in [MS-NRTP] section 2.2.2.8) in the following cases:

- The specified Server Type is not available.

- The specified constructor is not available.
- The values in the **__Args** field cannot be used to call the specified constructor.

<11> Section 3.3.3: Windows allows InitialLeaseTime, RenewOnCallTime and SponsorCallTimeout values to be overridden by the higher layer. The default values are as follows:

Property	Value
InitialLeaseTime	10 minutes
RenewOnCallTime	2 minutes
SponsorshipTimeout	2 minutes

8 Index

N

O

P

R

S

T

U

V

W