[MS-TSRV]: Timer Service Protocol

Revision Summary

Date	Revision History	Revision Class	Comments
05/22/2009	0.1	Major	First Release.
07/02/2009	0.1.1	Editorial	Revised and edited the technical content.
08/14/2009	0.1.2	Editorial	Revised and edited the technical content.
09/25/2009	0.2	Minor	Updated the technical content.

Contents

1 Introduction

This specification describes the Timer Service (TSRV) Protocol, a SOAP-based protocol that supports client applications which require a notification from an external service when a specified period of time has elapsed. TSRV allows a client to request a timer, request timer removal, and receive notices from the TimerService. The TimerService registers timers, sends expiration notices to the client, and provides the facilities for timer removal.

1.1 Glossary

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

globally unique identifier (GUID)
SOAP action
SOAP body
SOAP fault
SOAP header block
SOAP message
SOAP mustUnderstand attribute
Web Services Description Language (WSDL)
XML
XML namespace
XML schema

The following terms are specific to this document:

distributed denial-of-service (DDoS) attack: A DoS attack where multiple computers attempt to deny service on a single target computer.

registered timer: A client timer that has been registered with the server. The timer is registered after the server completes the REGISTER_TIMER event, which includes processing the client's Register Timer message and sending the Register Timer Response message.

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.

[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

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., and Winer, D., "Simple Object Access Protocol (SOAP) 1.1", May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

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

[WS-Addressing] Box, D., Christensen, E., Francisco, C., et al., "Web Services Addressing (WS-Addressing)", W3C Member Submission, August 2004, http://www.w3.org/Submission/ws-addressing/

[WS-Addr-Core] World Wide Web Consortium, "Web Services Addressing 1.0 - Core", W3C Recommendation, May 2006, http://www.w3.org/TR/ws-addr-core/

[WSASB] Gudgin, M., Hadley, M., and Rogers, T., "Web Services Addressing 1.0 - SOAP Binding", W3C Recommendation, May 2006, http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/

[WSDL] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1", W3C Note, March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[XMLNS] World Wide Web Consortium, "Namespaces in XML 1.0 (Second Edition)", August 2006, http://www.w3.org/TR/REC-xml-names/

[XMLSCHEMA1] Thompson, H.S., Ed., Beech, D., Ed., Maloney, M., Ed., and Mendelsohn, N., Ed., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

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

1.3 Protocol Overview (Synopsis)

TSRV allows a client to register a timer, remove a timer, and receive notification that a timer has expired. Two software components participate in this protocol.

Client: A software component that registers, removes, and receives notification of an expired timer.

TimerService: A software component that handles client requests to register and remove timers. The timer service is also responsible for notifying the **Callback Address**, which is specified during registration, that the timer has expired.

The diagram that follows illustrates the sequence of TSRV messages.

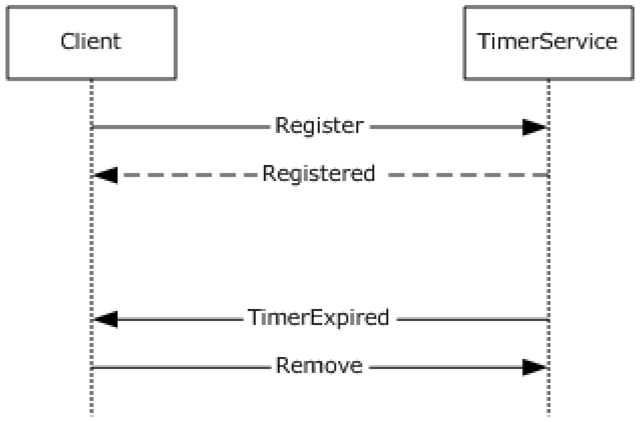

Figure 1: Protocol message sequence diagram

1.4 Relationship to Other Protocols

TSRV uses Web Services Addressing (W-S Addressing) [WS-Addressing] over SOAP [SOAP1.1], as shown in the following layering diagram.

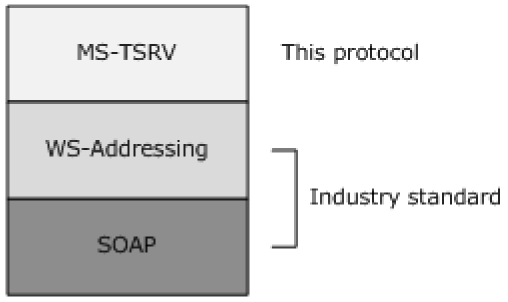

Figure 2: Protocol layering diagram

1.5 Prerequisites/Preconditions

TSRV has the following preconditions:

- The underlying protocol MUST support SOAP [SOAP].
- The client MUST already be configured with the server endpoint.
- The client **Callback Address** configuration MUST match that of the server.
- All clients MUST be able to communicate with the server.
- The server MUST be able to communicate with the client.

1.6 Applicability Statement

TSRV can be used to provide notification to the client that a certain period of time has elapsed. The address of the client to which notifications are sent is specified during timer registration.

1.7 Versioning and Capability Negotiation

This specification covers the following versioning issues.

Supported Transports: This protocol MUST be implemented by using a transport that supports SOAP HTTP Request and Response messages, as specified in [SOAP 1.1].

Protocol Versions: This protocol requires SOAP 1.1 [SOAP1.1] or SOAP 1.2 [SOAP1.2].

Localization: This protocol includes text strings in the Register Timer Fault message.

Capability Negotiation: This protocol does not support negotiation of the version to use.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

TSRV can be used over any transport protocol that supports the message transmission process that is specified by the following protocols.

- [SOAP1.1]
- [SOAP1.2]

This specification uses the term SOAP to mean either [SOAP1.1] or [SOAP1.2]. Where the differences between the two versions of the SOAP protocol are significant, either [SOAP1.1] or [SOAP1.2] is specified.

2.2 Common Message Syntax

2.2.1 Namespaces

This specification defines and references various **XML namespaces** that use the mechanisms specified in [XMLNS]. Although this specification associates a specific XML namespace prefix for each XML namespace that is used, the choice of any particular XML namespace prefix is implementation-specific and not significant for interoperability.

Prefix	Namespace URI	Reference
wsa10	http://www.w3.org/2005/08/addressing	[WS-Addr-Core] [WSASB]
env	http://schemas.xmlsoap.org/soap/envelope/	[SOAP1.1] [SOAP1.2/1]
wsan	http://schemas.microsoft.com/ws/2005/05/addressing/none	[WS-Addressing]
ts	http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService	[MS-TSRV]
soap	http://schemas.xmlsoap.org/wsdl/soap	[SOAP1.1] [SOAP1.2/1]
xsd	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1] [XMLSCHEMA2]
wsdl	http://schemas.xmlsoap.org/wsdl/	[WSDL]

2.2.2 Message Types

The following table summarizes the message types that are supported by TSRV.

Message	Sender	Description
Register Timer	Client	The Register Timer message is a request message that is used to register a new timer.
Register Timer	Server	The Register Timer Response message is a reply message that is used to

Message	Sender	Description
Response		register a new timer.
Timer Expired Notification	Server	The Timer Expired Notification message is a request message that is used to notify a **Callback Address** that a **registered timer** has expired.
Remove Timer	Client	The Remove Timer message is a request message that is used to remove a registered timer.
Register Timer Fault	Server	The Register Timer Fault message is a response message that contains a **SOAP fault**.

2.2.2.1 Register Timer Message

A Register Timer message request is a **SOAP message** that has the following additional constraints:

- A Register Timer message MUST include a **SOAP action** in the **SOAP header block** (as specified in [WS-Addressing] section 3), with the value set to http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/RegisterTimer.
- A Register Timer message MUST include a <RegisterTimer> element in the **SOAP body** that MUST specify values for the <duration> and <callbackEndpoint> elements, as shown in section 4.1.

2.2.2.2 Register Timer Response Message

A Register Timer Response message reply is a SOAP message that has the following additional constraints:

- A Register Timer Response message MUST include a SOAP action in the SOAP header block (as specified in [WS-Addressing] section 3), with the value set to http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/Registered.
- A Register Timer Response message MUST include a <RegisterTimerResponse> element in the SOAP body that MUST provide a value for the <RegisterTimerResult> element, as shown in section 4.2.

2.2.2.3 Timer Expired Notification Message

A Timer Expired Notification message request is a SOAP message that has the following additional constraints:

- A Timer Expired Notification message MUST include a SOAP action in the SOAP header block (as specified in [WS-Addressing] section 3), with the value set to http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerExpiredNotification/TimerExpiredNotification.
- A Timer Expired Notification message MUST include a <TimerExpiredNotification> element in the SOAP body and MUST provide a value for the <timerId> element, as shown in section 4.3.

2.2.2.4 Remove Timer Message

A Remove Timer message request is a SOAP message that has the following additional constraints:

- A Remove Timer message MUST include a SOAP action in the SOAP header block (as specified in [WS-Addressing] section 3), with the value set to http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/RemoveTimer.
- A Remove Timer message MUST include a <RemoveTimer> element in the SOAP body and MUST specify a value for the <timerId> element, as shown in section 4.4.

2.2.2.5 Register Timer Fault Message

A Register Timer Fault message reply is a SOAP message that contains a SOAP fault.

2.2.3 Elements

The following table summarizes the set of common **XML schema** element definitions that are defined by this specification. XML schema element definitions that are specific to a particular operation are described with the operation.

Element	Description
<timerId>	A **GUID** that identifies a registered timer.

2.2.3.1 timerId

The <timerId> element MUST include a GUID that identifies a registered timer.

2.2.4 Complex Types

This specification does not define any common XML schema complex types.

2.2.5 Simple Types

This specification does not define any common XML schema simple types.

2.2.6 Attributes

This specification does not define any common XML schema attributes.

2.2.7 Groups

This specification does not define any common XML schema groups.

2.2.8 Attribute Groups

This specification does not define any common XML schema attribute groups.

3 Protocol Details

3.1 Server Details

3.1.1 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 the behavior described in this document.

The TimerService has the following states, as illustrated in the figure:

- Start
- Registered
- TimerExpired
- Final

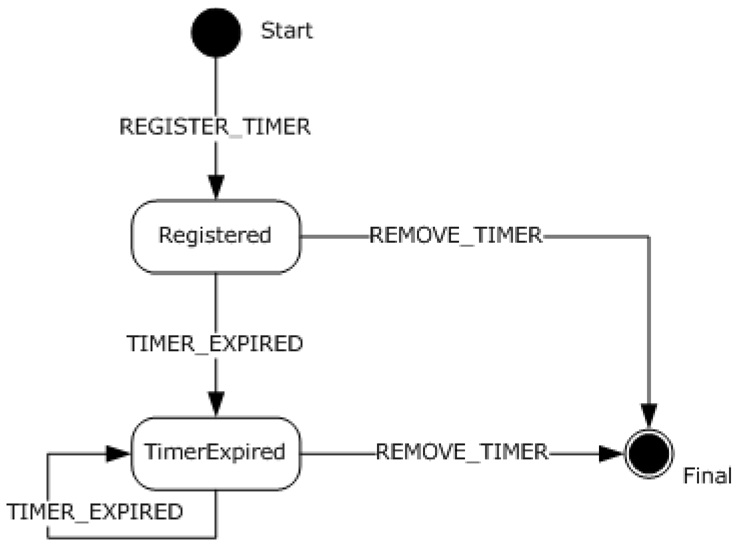

Figure 3: TimerService state machine diagram

The TimerService MUST maintain the following data fields for each timer:

Timer ID: A GUID that identifies the timer.

Expiration Time: A value that represents the time when the timer expires.

Number of Attempts: A non-negative integer value that represents the number of times the Timer Expired Notification message was sent.

Maximum Number of Attempts: A positive integer that represents the number of attempts to notify the client of an expired timer before transitioning to the Final state.

Callback Address: A WS-Addressing [WS-Addressing] Version 1 address that specifies the client address to be notified when the expiration time is reached.

3.1.1.1 Start State

The REGISTER_TIMER message processing event is processed in the Start state.

3.1.1.2 Registered State

The following events are processed in the Registered state:

- The REMOVE_TIMER message processing event.
- The TIMER_EXPIRED timer event.

3.1.1.3 TimerExpired State

The following events are processed in the TimerExpired state:

- The REMOVE_TIMER message processing event.
- The TIMER_EXPIRED timer event.

3.1.1.4 Final State

No events are triggered in the Final state.

3.1.2 Timers

TSRV defines the following timer on the server:

Expiration timer: The Expiration timer MUST trigger the TIMER_EXPIRED event.

3.1.3 Initialization

During initialization of the server, the following applies:

- The server MUST be in the Start state.
- A value for the **Maximum Number of Attempts** field (section 3.1.1) MUST be set.

3.1.4 Message Processing Events and Sequencing Rules

3.1.4.1 REGISTER_TIMER Event

The REGISTER_TIMER event MUST be signaled with the following argument:

Register Timer: The Register Timer message.

If the REGISTER_TIMER event is signaled, the TimerService MUST perform the following actions:

- If the TimerService is in the Start state, the following MUST occur:
  - The **Timer ID** field (section 3.1.1) MUST be set to a new GUID.

- The **Expiration Time** field MUST be set to the value of the <duration> element (section 3.1.4.1.2.2) in the <RegisterTimer> element (section 3.1.4.1.2.1) of the body of the Register Timer message, as specified in section 4.1. The value of the **Expiration Time** field SHOULD be validated.
    - If the value of the specified **Expiration Time** field is invalid, send the Register Timer Fault message, stop processing, and transition to the Final state.
- The **Callback Endpoint** field MUST be set to the value of the <callbackEndpoint> element (section 3.1.4.1.2.3) in the <RegisterTimer> element (section 3.1.4.1.2.1) of the body of the Register Timer message, as specified in section 4.1. The value of the **Callback Endpoint** field SHOULD be validated.
    - If the value of the specified **Callback Endpoint** field is invalid, send the Register Timer Fault message, stop processing, and transition to the Final state.
- The value of the **Number of Attempts** field (section 3.1.1) MUST be set to zero.
- The expiration timer MUST be started and the interval at which it fires MUST be set to the **Expiration Time** field value.
- Transition to the Registered state.
- Send the Register Timer Response message with the <RegisterTimerResult> element (section 4.2) set to the value of the **Timer ID** field (section 3.1.1).

- Otherwise:
    - Do nothing.

3.1.4.1.1 Messages

The following logical relationships MUST exist between the REGISTER_TIMER event and the specified messages:

Register Timer message: The REGISTER_TIMER event MUST be triggered by a Register Timer message.

Register Timer Response message: The TimerService MUST respond to the REGISTER_TIMER event with the Register Timer Response message if the <duration> element (section 3.1.4.1.2.2) and <callbackEndpoint> element (section 3.1.4.1.2.3) are valid.

Register Timer Fault message: The TimerService MUST respond to the REGISTER_TIMER event with the Register Timer Fault message if either the <duration> element (section 3.1.4.1.2.2) or <callbackEndpoint> element (section 3.1.4.1.2.3) is invalid.

3.1.4.1.2 Elements

The following XML schema element definitions are specific to processing the REGISTER_TIMER event.

3.1.4.1.2.1 RegisterTimer Element

A complex element that consists of the <duration> element (section 3.1.4.1.2.2) and <callbackEndpoint> element (section 3.1.4.1.2.3).

3.1.4.1.2.2 duration Element

The value of the <duration> element defines the time span for the timer to be registered.

3.1.4.1.2.3 callbackEndpoint Element

When the timer expires, the value of the <callbackEndpoint> element defines the WS-Addressing [WS-Addressing] endpoint of the client to be notified.

3.1.4.1.3 Complex Types

The REGISTER_TIMER event does not define any complex types.

3.1.4.1.4 Simple Types

The REGISTER_TIMER event does not define any simple types.

3.1.4.2 REMOVE_TIMER Event

This event MUST be triggered by the following argument:

TimerId**:** A GUID that identifies the timer to remove.

If the REMOVE_TIMER event is triggered, and the TimerService is in the Registered or TimerExpired state, then transition to the Final state.

3.1.4.2.1 Messages

The following logical relationship applies to the Remove Timer message and the REMOVE_TIMER event:

- The Remove Timer message MAY trigger the REMOVE_TIMER event.

3.1.4.2.2 Elements

In this section, the XML schema element definitions are specific to processing the REMOVE_TIMER event.

3.1.4.2.2.1 RemoveTimer Element

The RemoveTimer element is a complex element that contains the <timerId> element (section 2.2.3.1).

3.1.4.2.2.2 TimerId Element

The value of the <timerId> element is a GUID that identifies the timer to remove.

3.1.4.2.3 Complex Types

The REMOVE_TIMER event does not define any complex types.

3.1.4.2.4 Simple Types

The REMOVE_TIMER event does not define any simple types.

3.1.5 Timer Events

When the expiration timer fires, it triggers the TIMER_EXPIRED event.

3.1.5.1 TIMER_EXPIRED Event

If the TIMER_EXPIRED event is triggered, the TimerService MUST perform the following actions:

- The **Number of Attempts** field (section 3.1.1) MUST be incremented by one.
- Send the Timer Expired Notification message on the underlying protocol to the address that is specified by the **Callback Address** field (section 3.1.1), while passing the GUID value of the **Timer ID** field (section 3.1.1).
- If the **Number of Attempts** field (section 3.1.1) exceeds the **Maximum Number of Attempts** field (section 3.1.1), trigger the REMOVE_TIMER event.
- Transition to the TimerExpired state.

3.1.6 Other Local Events

None.

4 Protocol Examples

4.1 Register Timer Message

The following example shows a Register Timer message that is sent by a client to request the registration of a timer that has a 1-hour expiration.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header>
<Action env:mustUnderstand="1"
xmlns:wsan="http://schemas.microsoft.com/ws/2005/05/addressing/none">
http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/RegisterTimer
</Action>
</env:Header>
<env:Body>
<RegisterTimer xmlns:ts="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService">
<duration>
PT30S
</duration>
<callbackEndpoint>
<Address xmlns:wsa10="http://www.w3.org/2005/08/addressing">
http://localhost/Client/TimerExpired
</Address>
</callbackEndpoint>
</RegisterTimer>
</env:Body>
</env:Envelope>
```

4.2 Register Timer Response Message

The following example shows a Register Timer Response message that is sent by the TimerService. This message specifies the GUID that identifies the registered timer.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header>
<Action env:mustUnderstand="1"
xmlns:wsan="http://schemas.microsoft.com/ws/2005/05/addressing/none">
http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/Registered
</Action>
</env:Header>
<env:Body>
<RegisterTimerResponse
xmlns:ts="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService">
<RegisterTimerResult>
49cb55e4-969e-4efd-a194-da227cc7ad7e
</RegisterTimerResult>
</RegisterTimerResponse>
</env:Body>
</env:Envelope>
```

4.3 Timer Expired Notification Message

The following example shows a Timer Expired Notification message that is sent by the TimerService. The message provides the GUID that identifies the expired timer.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Header>
<Action s:mustUnderstand="1" xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">
http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerExpiredNotification/
TimerExpiredNotification
</Action>
</s:Header>
<s:Body>
<TimerExpiredNotification xmlns="http://schemas.microsoft.com/netfx/2009/02/Timer/
ITimerExpiredNotification">
<timerId>
4f89ff76-7c1a-4e4c-a551-57f71d1e35af
</timerId>
</TimerExpiredNotification>
</s:Body>
</s:Envelope>
```

4.4 Remove Timer Message

The following example shows a Remove Timer message that is sent by a client. The message provides a GUID that identifies the expired timer to remove.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header>
<Action env:mustUnderstand="1"
xmlns:wsan="http://schemas.microsoft.com/ws/2005/05/addressing/none">
http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/RemoveTimer
</Action>
</env:Header>
<env:Body>
<RemoveTimer xmlns:ts="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService">
<timerId>
49cb55e4-969e-4efd-a194-da227cc7ad7e
</timerId>
</RemoveTimer>
</env:Body>
</env:Envelope>
```

4.5 Register Timer Fault Message

The following example shows a Register Timer Fault message that contains fault details. The message is sent by the TimerService if an invalid value exists for either the **Expiration Time** of the timer (section 3.1.1) or the **Callback Address** of the client (section 3.1.1).

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
```

```
<env:Header>
<Action env:mustUnderstand="1"
xmlns:wsan="http://schemas.microsoft.com/ws/2005/05/addressing/none">
http://schemas.microsoft.com/net/2005/12/windowscommunicationfoundation/
dispatcher/fault
</Action>
</env:Header>
<env:Body>
<env:Fault>
<faultcode xmlns:wcf="http://schemas.microsoft.com/net/2005/12/
windowscommunicationfoundation/dispatcher">
wcf:InternalServiceFault
</faultcode>
 <faultstring xml:lang="en-US">
The server was unable to process the request due to an internal error.  For more information
about the error, either turn on IncludeExceptionDetailInFaults (either from
ServiceBehaviorAttribute or from the &lt;serviceDebug&gt; configuration behavior) on the
server in order to send the exception information back to the client, or turn on tracing as
per the Microsoft .NET Framework 3.0 SDK documentation and inspect the server trace logs.
</faultstring>
</env:Fault>
</env:Body>
</env:Envelope>
```

5 Security

5.1 Security Considerations for Implementers

TSRV security is defined by the underlying protocol implementation (section 1.4). The implementer should address the following security considerations when implementing the server side of TSRV:

Wire traffic security: Although wire traffic depends on the underlying protocol implementation, passed data, such as the <callbackEndPoint> address (section 3.1.4.1.2.3), the <timerId> value (section 2.2.3.1), and exception information, can be maliciously used. The implementer should use an underlying protocol that secures TSRV messages.

Distributed denial-of-service attack (DDoS) attacks: This security concern involves the potential for the TimerService to be used to conduct **DDoS attacks**. The implementer may verify that the requested **Callback Address** is the same as the originating client address.

5.2 Index of Security Parameters

None.

6 Appendix A: Full WSDL

For ease of implementation the full WSDL is provided below:

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ts="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:q1="http://schemas.datacontract.org/2004/07/System.ServiceModel.Activities">
<wsdl:types>
<xsd:schema
targetNamespace="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/Imports">
<xsd:import namespace="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService" />
<xsd:import
namespace="http://schemas.datacontract.org/2004/07/System.ServiceModel.Activities" />
<xsd:import
namespace="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerExpiredNotification" />
<xsd:import namespace="http://schemas.microsoft.com/2003/10/Serialization/" />
<xsd:import namespace="http://www.w3.org/2005/08/addressing" />
<xsd:import namespace="http://schemas.datacontract.org/2004/07/System.ServiceModel" />
<xsd:import namespace="http://schemas.datacontract.org/2004/07/System" />
</xsd:schema>
</wsdl:types>
<wsdl:message name="ITimerService_RegisterTimer_InputMessage">
<wsdl:part name="parameters" element="ts:RegisterTimer" />
</wsdl:message>
<wsdl:message name="ITimerService_RegisterTimer_OutputMessage">
<wsdl:part name="parameters" element="ts:RegisterTimerResponse" />
</wsdl:message>
<wsdl:message name="ITimerService_RegisterTimer_TimerExceptionFault_FaultMessage">
<wsdl:part name="detail" element="q1:TimerException" />
</wsdl:message>
<wsdl:message name="ITimerService_RemoveTimer_InputMessage">
<wsdl:part name="parameters" element="ts:RemoveTimer" />
</wsdl:message>
<wsdl:portType name="ITimerService">
<wsdl:operation name="RegisterTimer">
<wsdl:input
wsaw:Action="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/RegisterTimer"
message="ts:ITimerService_RegisterTimer_InputMessage" />
<wsdl:output
wsaw:Action="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/Registered"
message="ts:ITimerService_RegisterTimer_OutputMessage" />
<wsdl:fault
wsaw:Action="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/ITimerService/Reg
isterTimerTimerExceptionFault" name="TimerExceptionFault"
message="ts:ITimerService_RegisterTimer_TimerExceptionFault_FaultMessage" />
</wsdl:operation>
<wsdl:operation name="RemoveTimer">
<wsdl:input
wsaw:Action="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/RemoveTimer"
message="ts:ITimerService_RemoveTimer_InputMessage" />
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="DefaultBinding_ITimerService" type="ts:ITimerService">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="RegisterTimer">
```

```
<soap:operation
soapAction="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/RegisterTimer"
style="document" />
<wsdl:input>
<soap:body use="literal" />
</wsdl:input>
<wsdl:output>
<soap:body use="literal" />
</wsdl:output>
<wsdl:fault name="TimerExceptionFault">
<soap:fault name="TimerExceptionFault" use="literal" />
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="RemoveTimer">
<soap:operation
soapAction="http://schemas.microsoft.com/netfx/2009/02/Timer/ITimerService/RemoveTimer"
style="document" />
<wsdl:input>
<soap:body use="literal" />
</wsdl:input>
</wsdl:operation>
</wsdl:binding>
</wsdl:definitions>
```

7 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products:

- Microsoft® .NET Framework Version 4.0

Exceptions, if any, are noted below. If a service pack number appears with the product version, behavior changed in that service pack. The new behavior also applies to subsequent service packs of the product unless otherwise specified.

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.

8 Change Tracking

This section identifies changes made to [MS-TSRV] protocol documentation between August 2009 and September 2009 releases. Changes are classed as major, minor, or editorial.

Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- A protocol is deprecated.
- The removal of a document from the documentation set.
- Changes made for template compliance.

Minor changes do not affect protocol interoperability or implementation. Examples are updates to fix technical accuracy or ambiguity at the sentence, paragraph, or table level.

Editorial changes apply to grammatical, formatting, and style issues.

No changes means that the document is identical to its last release.

Major and minor changes can be described further using the following revision types:

- New content added.
- Content update.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.

- Content removed for template compliance.
- Obsolete document removed.

Editorial changes always have the revision type "Editorially updated."

Some important terms used in revision type descriptions are defined as follows:

Protocol syntax refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.

Protocol revision refers to changes made to a protocol that affect the bits that are sent over the wire.

Changes are listed in the following table. If you need further information, please contact protocol@microsoft.com.

Section	**Tracking number (if applicable) and description**	**Major change (Y or N)**	**Revision Type**
1.2.1 Normative References	Moved reference for [MS-GLOS] to Informative References.	N	Editorially updated.

9 Index