

[MS-ALERTSS]:

Alerts Service Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation (“this documentation”) for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](#) or the [Microsoft Community Promise](#). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](#).
- **Trademarks.** The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation 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. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1	New	Initial Availability
6/27/2008	1.0	Major	Revised and edited the technical content
12/12/2008	1.01	Editorial	Revised and edited the technical content
7/13/2009	1.02	Major	Revised and edited the technical content
8/28/2009	1.03	Editorial	Revised and edited the technical content
11/6/2009	1.04	Editorial	Revised and edited the technical content
2/19/2010	2.0	Minor	Updated the technical content
3/31/2010	2.01	Editorial	Revised and edited the technical content
4/30/2010	2.02	Editorial	Revised and edited the technical content
6/7/2010	2.03	Editorial	Revised and edited the technical content
6/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
7/23/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	2.04	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	2.04	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	3.0	Major	Significantly changed the technical content.
4/11/2012	3.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	3.0	None	No changes to the meaning, language, or formatting of the technical content.
9/12/2012	3.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	3.1	Minor	Clarified the meaning of the technical content.
2/11/2013	3.1	None	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	4.0	Major	Significantly changed the technical content.
11/18/2013	4.0	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
2/10/2014	4.0	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	4.0	None	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	4.0	None	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	4.0	None	No changes to the meaning, language, or formatting of the technical content.
2/26/2016	5.0	Major	Significantly changed the technical content.
7/15/2016	5.0	None	No changes to the meaning, language, or formatting of the technical content.
9/14/2016	5.0	None	No changes to the meaning, language, or formatting of the technical content.
9/19/2017	5.1	Minor	Clarified the meaning of the technical content.
7/24/2018	6.0	Major	Significantly changed the technical content.
10/1/2018	7.0	Major	Significantly changed the technical content.
6/18/2019	7.0	None	No changes to the meaning, language, or formatting of the technical content.
7/20/2021	8.0	Major	Significantly changed the technical content.
10/5/2021	9.0	Major	Significantly changed the technical content.

Table of Contents

1	Introduction	6
1.1	Glossary	6
1.2	References	7
1.2.1	Normative References	8
1.2.2	Informative References	8
1.3	Protocol Overview (Synopsis)	8
1.3.1	Alert Enumeration.....	9
1.3.2	Alert Deletion.....	9
1.4	Relationship to Other Protocols	9
1.5	Prerequisites/Preconditions	10
1.6	Applicability Statement	10
1.7	Versioning and Capability Negotiation	10
1.8	Vendor-Extensible Fields	10
1.9	Standards Assignments.....	10
2	Messages.....	11
2.1	Transport.....	11
2.2	Common Message Syntax	11
2.2.1	Namespaces	11
2.2.2	Messages.....	11
2.2.3	Elements	11
2.2.4	Complex Types.....	11
2.2.5	Simple Types	12
2.2.6	Attributes	12
2.2.7	Groups	12
2.2.8	Attribute Groups.....	12
3	Protocol Details.....	13
3.1	Alerts Service Protocol Server Details	13
3.1.1	Abstract Data Model.....	13
3.1.2	Timers	13
3.1.3	Initialization.....	13
3.1.4	Message Processing Events and Sequencing Rules	13
3.1.4.1	DeleteAlerts	14
3.1.4.1.1	Messages	14
3.1.4.1.1.1	DeleteAlertsSoapIn	14
3.1.4.1.1.2	DeleteAlertsSoapOut	14
3.1.4.1.2	Elements.....	14
3.1.4.1.2.1	DeleteAlerts	14
3.1.4.1.2.2	DeleteAlertsResponse.....	15
3.1.4.1.3	Complex Types	15
3.1.4.1.3.1	ArrayOfString	15
3.1.4.1.3.2	ArrayOfDeleteFailureDefinition.....	15
3.1.4.1.3.3	DeleteFailureDefinition	16
3.1.4.1.4	Simple Types	16
3.1.4.1.4.1	ErrorType	16
3.1.4.2	GetAlerts	16
3.1.4.2.1	Messages	17
3.1.4.2.1.1	GetAlertsSoapIn	17
3.1.4.2.1.2	GetAlertsSoapOut	17
3.1.4.2.2	Elements.....	17
3.1.4.2.2.1	GetAlerts	17
3.1.4.2.2.2	GetAlertsResponse.....	17
3.1.4.2.3	Complex Types	17
3.1.4.2.3.1	AlertInfoDefinition.....	18

3.1.4.2.3.2	ArrayOfAlertDefinition	18
3.1.4.2.3.3	Alert.....	19
3.1.4.2.3.4	ArrayOfDeliveryChannelDefinition	19
3.1.4.2.3.5	DeliveryChannel	20
3.1.4.2.3.6	EmailChannel	20
3.1.5	Timer Events.....	20
3.1.6	Other Local Events.....	20
4	Protocol Examples	21
5	Security	23
5.1	Security Considerations for Implementers	23
5.2	Index of Security Parameters	23
6	Appendix A: Full WSDL	24
7	Appendix B: Product Behavior	27
8	Change Tracking.....	28
9	Index.....	29

1 Introduction

The Alerts Service Protocol allows a protocol client to list and delete alert subscriptions. Alert subscriptions specify when and how notifications are sent to users when changes are made to content stored on the server. The protocol does not specify the creation or editing of alert subscriptions.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

alert: An Internet message that is sent to subscribers automatically to notify them when user-defined criteria are met. Alerts are generated automatically when items such as documents, webpages, list items, sites, or other resources on a server are changed.

alert subscription: A request to receive an Internet message automatically when user-defined criteria are met. Such messages are generated automatically when items such as documents, webpages, list items, sites, or other resources on a server are changed.

authenticated user: A built-in security group specified in [\[MS-WSO\]](#) whose members include all users that can be authenticated by a computer.

current user: The user who is authenticated during processing operations on a front-end web server or a back-end database server.

document: An object in a content database such as a file, folder, **list**, or **site**. Each object is identified by a **URI**.

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [\[RFC4122\]](#) or [\[C706\]](#) must be used for generating the **GUID**. See also universally unique identifier (UUID).

Hypertext Transfer Protocol (HTTP): An application-level protocol for distributed, collaborative, hypermedia information systems (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

Hypertext Transfer Protocol Secure (HTTPS): An extension of HTTP that securely encrypts and decrypts web page requests. In some older protocols, "Hypertext Transfer Protocol over Secure Sockets Layer" is still used (Secure Sockets Layer has been deprecated). For more information, see [\[SSL3\]](#) and [\[RFC5246\]](#).

item: A unit of content that can be indexed and searched by a search application.

list: A container within a SharePoint site that stores list items. A list has a customizable schema that is composed of one or more fields.

list item: An individual entry within a SharePoint list. Each list item has a schema that maps to fields in the list that contains the item, depending on the content type of the item.

site: A group of related pages and data within a SharePoint site collection. The structure and content of a site is based on a site definition. Also referred to as SharePoint site and web site.

SOAP: A lightweight protocol for exchanging structured information in a decentralized, distributed environment. **SOAP** uses XML technologies to define an extensible messaging framework, which

provides a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation-specific semantics. SOAP 1.2 supersedes SOAP 1.1. See [\[SOAP1.2-1/2003\]](#).

SOAP action: The HTTP request header field used to indicate the intent of the **SOAP** request, using a **URI** value. See [\[SOAP1.1\]](#) section 6.1.1 for more information.

SOAP body: A container for the payload data being delivered by a SOAP message to its recipient. See [\[SOAP1.2-1/2007\]](#) section 5.3 for more information.

SOAP fault: A container for error and status information within a SOAP message. See [\[SOAP1.2-1/2007\]](#) section 5.4 for more information.

Uniform Resource Identifier (URI): A string that identifies a resource. The URI is an addressing mechanism defined in Internet Engineering Task Force (IETF) Uniform Resource Identifier (URI): Generic Syntax [\[RFC3986\]](#).

Uniform Resource Locator (URL): A string of characters in a standardized format that identifies a document or resource on the World Wide Web. The format is as specified in [\[RFC1738\]](#).

web discussion comment: An individual comment that is added to a web discussion.

Web Services Description Language (WSDL): An XML format for describing network services as a set of endpoints that operate on messages that contain either document-oriented or procedure-oriented information. The operations and messages are described abstractly and are bound to a concrete network protocol and message format in order to define an endpoint. Related concrete endpoints are combined into abstract endpoints, which describe a network service. WSDL is extensible, which allows the description of endpoints and their messages regardless of the message formats or network protocols that are used.

WSDL operation: A single action or function of a web service. The execution of a WSDL operation typically requires the exchange of messages between the service requestor and the service provider.

WSDL port type: A named set of logically-related, abstract **Web Services Description Language (WSDL)** operations and messages.

XML namespace: A collection of names that is used to identify elements, types, and attributes in XML documents identified in a URI reference [\[RFC3986\]](#). A combination of XML namespace and local name allows XML documents to use elements, types, and attributes that have the same names but come from different sources. For more information, see [\[XMLNS-2ED\]](#).

XML schema: A description of a type of XML document that is typically expressed in terms of constraints on the structure and content of documents of that type, in addition to the basic syntax constraints that are imposed by XML itself. An XML schema provides a view of a document type at a relatively high level of abstraction.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](#).

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.

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, November 1987, <http://www.ietf.org/rfc/rfc1035.txt>

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

[RFC2396] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998, <http://www.rfc-editor.org/rfc/rfc2396.txt>

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, <http://www.rfc-editor.org/rfc/rfc2616.txt>

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, <http://www.rfc-editor.org/rfc/rfc2818.txt>

[RFC2822] Resnick, P., Ed., "Internet Message Format", RFC 2822, April 2001, <http://www.ietf.org/rfc/rfc2822.txt>

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., et al., "Simple Object Access Protocol (SOAP) 1.1", W3C Note, May 2000, <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

[SOAP1.2-1/2007] Gudgin, M., Hadley, M., Mendelsohn, N., et al., "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)", W3C Recommendation, April 2007, <http://www.w3.org/TR/2007/REC-soap12-part1-20070427/>

[SOAP1.2-2/2007] Gudgin, M., Hadley, M., Mendelsohn, N., et al., "SOAP Version 1.2 Part 2: Adjuncts (Second Edition)", W3C Recommendation, April 2007, <http://www.w3.org/TR/2007/REC-soap12-part2-20070427>

[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] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, <http://www.w3.org/TR/2009/REC-xml-names-20091208/>

[XMLSCHEMA1] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "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-OSALER] Microsoft Corporation, "[Alerts Interoperability Protocol](#)".

1.3 Protocol Overview (Synopsis)

The protocol allows a protocol client to list and delete existing **alert subscriptions**. It consists of a single **WSDL port type** with two **WSDL operations** and their replies. It describes a series of communications between the protocol client and protocol server roles.

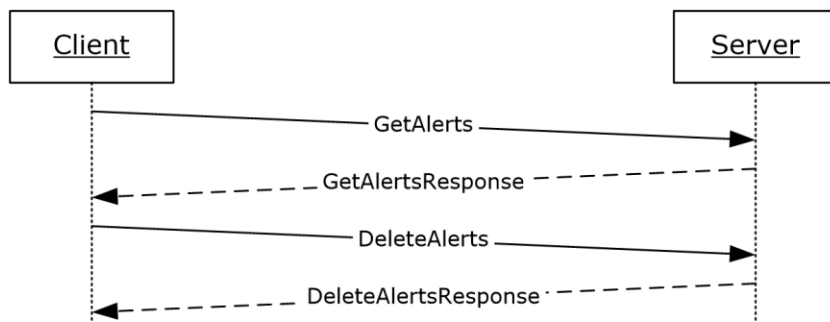


Figure 1: Protocol client/server message exchange

Prior to the initiation of the protocol, the protocol client is configured with user information. After this is done, the client connects by using the user information. For more information, see section [1.5](#).

1.3.1 Alert Enumeration

The protocol client can now initiate an exchange by requesting a list of alert subscriptions for the specified user. The server responds with a list of metadata about the alert subscriptions to which the **authenticated user** has subscribed.

1.3.2 Alert Deletion

The protocol client can delete an alert subscription by sending a request to the protocol server that contains a list of **GUIDs** for the alert subscriptions to delete. The protocol server responds with information about deletion(s) that failed. The protocol server will no longer send the alert subscriptions for the GUIDs that did not fail.

1.4 Relationship to Other Protocols

This protocol uses the **SOAP** message protocol for formatting request and response messages, as described in [\[SOAP1.1\]](#), [\[SOAP1.2-1/2007\]](#) and [\[SOAP1.2-2/2007\]](#). It transmits those messages by using **HTTP**, as described in [\[RFC2616\]](#), or **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)**, as described in [\[RFC2818\]](#).

The following diagram shows the underlying messaging and transport stack used by the protocol.

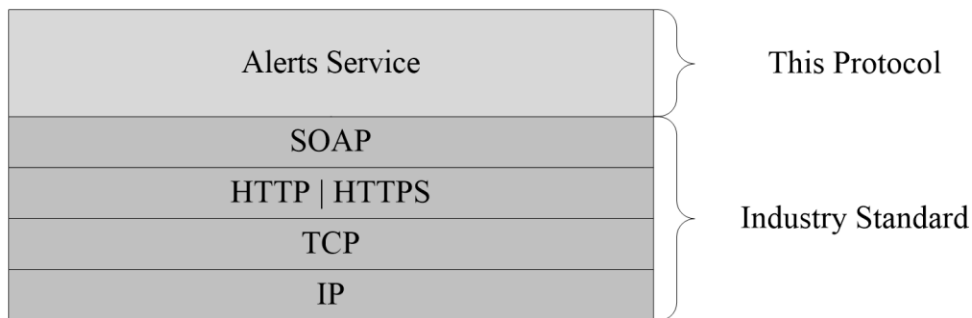


Figure 2: This protocol in relation to other protocols

The protocol has no interactions with parallel protocols, nor are there other protocols that substitute for it.

1.5 Prerequisites/Preconditions

This protocol operates against a **site** that is identified by a **URL** that is known by protocol clients. The protocol server endpoint is formed by appending "/_vti_bin/Alerts.asmx" to the URL of the site, for example, http://www.contoso.com/Repository/_vti_bin/Alerts.asmx.

This protocol assumes that authentication has been performed by the underlying protocols.

1.6 Applicability Statement

This protocol is used to assist in the management of **alert subscriptions**. The service allows a protocol client to get the list of alert subscriptions for a particular **authenticated user**. It also allows a protocol client to delete alert subscriptions.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports:** This protocol uses multiple transports with **SOAP** as specified in section [2.1](#).

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Protocol servers MUST support SOAP over HTTP. Protocol servers SHOULD additionally support SOAP over HTTPS, as specified in [\[RFC2818\]](#), for securing communication with protocol clients.

Protocol messages MUST be formatted as specified in [\[SOAP1.1\]](#), section 4, "SOAP Envelope", or in [\[SOAP1.2-1/2007\]](#), section 5, "SOAP Message Construct". Protocol server faults MUST be returned either by using HTTP Status Codes as specified in [\[RFC2616\]](#), section 10, "Status Code Definitions", or by using **SOAP faults** as specified in [\[SOAP1.1\]](#), section 4.4, "SOAP Fault" or in [\[SOAP1.2-1/2007\]](#), section 5.4, "SOAP Fault".

2.2 Common Message Syntax

This section contains common definitions used by this protocol. The syntax of the definitions uses **XML schema** as specified in [\[XMLSCHEMA1\]](#) and [\[XMLSCHEMA2\]](#), and **Web Services Description Language (WSDL)** as specified in [\[WSDL\]](#).

2.2.1 Namespaces

This specification defines and references various **XML namespaces** using 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
soap	http://schemas.xmlsoap.org/wsdl/soap/	[SOAP1.1]
tns	http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts/	
s	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1] [XMLSCHEMA2]
soap12	http://schemas.xmlsoap.org/wsdl/soap12/	[SOAP1.2-1/2007] [SOAP1.2-2/2007]
wsdl	http://schemas.xmlsoap.org/wsdl/	[WSDL]

2.2.2 Messages

This specification does not define any common WSDL message definitions.

2.2.3 Elements

This specification does not define any common XML Schema element definitions.

2.2.4 Complex Types

This specification does not define any common XML Schema complex type definitions.

2.2.5 Simple Types

This specification does not define any common XML Schema simple type definitions.

2.2.6 Attributes

This specification does not define any common XML Schema attribute definitions.

2.2.7 Groups

This specification does not define any common XML Schema group definitions.

2.2.8 Attribute Groups

This specification does not define any common XML Schema attribute group definitions.

3 Protocol Details

The client side of this protocol is a pass-through. No additional timers or other states are 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 allows protocol servers to notify protocol clients of application-level faults by using SOAP faults. Except where specified, these SOAP faults are not significant for interoperability, and protocol clients can interpret them in an implementation-specific manner.

This protocol allows protocol servers to perform implementation-specific authorization checks and notify protocol clients of authorization faults either by using HTTP Status Codes or by using SOAP faults as specified previously in this section.

3.1 Alerts Service Protocol Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model of data organization that an implementation maintains to participate in this protocol. 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.

Information about notifications and **alert subscriptions** is specified in [\[MS-OSALER\]](#).

This protocol retrieves and deletes alert subscriptions. Each **authenticated user** will have an individual list of alert subscriptions.

An alert subscription is a persistent request on the server for a notification (**alert**) that a particular **document** or **list** has been modified in a specific way. The alert subscription contains the information that determines when to generate a notification, and how to deliver the notification.

3.1.2 Timers

None.

3.1.3 Initialization

The initial protocol state is provided to the protocol by user authentication and connection to its server.

3.1.4 Message Processing Events and Sequencing Rules

The following table specifies the **WSDL** operations provided by the protocol.

Operation	Description
DeleteAlerts	Deletes alert subscriptions from a user's current list of alert subscriptions. Returns a DeleteAlertsResponse . If the server cannot locate an alert subscription using an identifier from the List, or cannot properly parse the identifier, it skips that identifier without notifying the client.
GetAlerts	Enumerates the current alert subscriptions for an authenticated user . It returns a GetAlertsResponse .

The protocol client is not required to perform a **GetAlerts** request prior to submitting a **DeleteAlerts** request. However, using the **GetAlerts** operation is the typical method for acquiring valid **AlertIDs** to provide to the **DeleteAlerts** operation.

3.1.4.1 DeleteAlerts

This operation deletes **alert subscriptions** from a user's current list of alert subscriptions.

```
<wsdl:operation name="DeleteAlerts">
  <wsdl:input message="tns:DeleteAlertsSoapIn" />
  <wsdl:output message="tns:DeleteAlertsSoapOut" />
</wsdl:operation>
```

The protocol client sends a **DeleteAlertsSoapIn** request message and the protocol server responds with a **DeleteAlertsSoapOut** response message.

The protocol client provides a list of alert identifiers in the **DeleteAlertsSoapIn** message. The protocol server deletes each alert subscription on the protocol server that matches one of the provided identifiers. The protocol server then responds with information about any deletions that failed. The response does not include information about successful deletions.

Deletion errors are counted during this operation. The maximum number of errors is implementation-dependent [<1>](#). If the maximum number of errors is reached, processing is terminated and a **DeleteFailure** error message that contains the **TooManyErrors** error type is appended to the response message.

If the string representing an identifier is not in the proper format, that identifier is ignored in the **DeleteAlertsSoapOut** response message, but the error is still counted toward the total number of deletion errors.

3.1.4.1.1 Messages

3.1.4.1.1.1 DeleteAlertsSoapIn

This message contains the request to begin a **DeleteAlerts WSDL** operation.

The **SOAP action** value of the message is defined as:

```
http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts/DeleteAlerts
```

The **SOAP body** contains a **DeleteAlerts** element.

3.1.4.1.1.2 DeleteAlertsSoapOut

This message represents the response associated with the **DeleteAlerts WSDL** operation. It contains the status for any part of the **DeleteAlerts** WSDL operation that failed. If the reply is empty, there were no failures of the error type specified in section [3.1.4.1.4.1](#).

The SOAP body contains a **DeleteAlertsResponse** element.

3.1.4.1.2 Elements

3.1.4.1.2.1 DeleteAlerts

This structure is contained in a **DeleteAlertsSoapIn** message and contains the list of **GUIDs** that the client is requesting to be deleted.

```
<s:element name="DeleteAlerts">
  <s:complexType>
    <s:sequence>
      <s:element name="IDs" type="tns:ArrayOfString" minOccurs="0" maxOccurs="1"/>
    </s:sequence>
  </s:complexType>
</s:element>
```

```
</s:complexType>
</s:element>
```

IDs: This is an **ArrayOfString** structure with one string for each alert subscription to be deleted. The string **MUST** be formatted as a GUID.

3.1.4.1.2.2 DeleteAlertsResponse

This structure is the content of a **DeleteAlertsSoapOut** message. It contains a list of **DeleteAlertsResult** structures, with at least one for each GUID in the **DeleteAlertsRequest** that failed during processing.

```
<s:element name="DeleteAlertsResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="DeleteAlertsResult"
        type="tns:ArrayOfDeleteFailureDefinition" minOccurs="0" maxOccurs="1"/>
    </s:sequence>
  </s:complexType>
</s:element>
```

DeleteAlertsResult: A structure of type **ArrayOfDeleteFailureDefinition**, as specified in **ArrayOfDeleteFailureDefinition** (section [3.1.4.1.3.2](#)).

3.1.4.1.3 Complex Types

3.1.4.1.3.1 ArrayOfString

In this protocol, an **ArrayOfString** structure is used to pass multiple strings, each of which represents a GUID associated with a specific alert subscription. Each string in the sequence **MUST** be formatted as a GUID.

```
<s:complexType name="ArrayOfString">
  <s:sequence>
    <s:element name="string" type="s:string" nillable="true" minOccurs="0"
      maxOccurs="unbounded"/>
  </s:sequence>
</s:complexType>
```

string: Each string in the array represents a single GUID which references a particular alert subscription.

3.1.4.1.3.2 ArrayOfDeleteFailureDefinition

This structure is a list of **DeleteFailure** structures, each of which represents a single failure while processing a **DeleteAlerts** operation.

```
<s:complexType name="ArrayOfDeleteFailureDefinition">
  <s:sequence>
    <s:element name="DeleteFailure"
      type="tns>DeleteFailureDefinition" minOccurs="0"
      maxOccurs="unbounded"/>
  </s:sequence>
</s:complexType>
```

DeleteFailure: A single **DeleteFailureDefinition** structure, as specified in section [3.1.4.1.3.3](#).

3.1.4.1.3.3 DeleteFailureDefinition

This structure represents the error message associated with a failure that occurred while processing one **GUID** from a **DeleteAlerts** operation. The structure contains the GUID that failed as a string, and an **ErrorType** indicating why the failure occurred.

```
<s:complexType name="DeleteFailureDefinition">
  <s:sequence>
    <s:element name="ID" type="s:string" minOccurs="0"/>
    <s:element name="Error" type="tns:ErrorType"/>
  </s:sequence>
</s:complexType>
```

ID: A string representing the GUID that failed. The string MUST be formatted as a GUID. This element is not returned when **Error** is **TooManyErrors**.

Error: The **ErrorType** indicating why the deletion failed.

3.1.4.1.4 Simple Types

3.1.4.1.4.1 ErrorType

ErrorType describes a particular failure that occurred during **DeleteAlerts** request processing.

```
<s:simpleType name="ErrorType">
  <s:restriction base="s:string">
    <s:enumeration value="None"/>
    <s:enumeration value="AccessDenied"/>
    <s:enumeration value="ServerError"/>
    <s:enumeration value="TooManyErrors"/>
  </s:restriction>
</s:simpleType>
```

The following table defines the allowable values. The protocol server MUST NOT return a value of "None".

Value	Meaning
None	The alert subscription was successfully deleted.
AccessDenied	An authenticated user is not authorized to delete the specified alert subscription. <2>
ServerError	An unknown or implementation-specific protocol server error occurred during a DeleteAlerts operation.
TooManyErrors	The maximum number of failures occurred while the protocol server was processing a DeleteAlerts request message.

3.1.4.2 GetAlerts

This operation lists the current alert subscriptions for an **authenticated user**.

```
<wsdl:operation name="GetAlerts">
  <wsdl:input message="tns:GetAlertsSoapIn" />
  <wsdl:output message="tns:GetAlertsSoapOut" />
</wsdl:operation>
```



```
</wsdl:operation>
```

The protocol client sends a **GetAlertsSoapIn** request message and the protocol server responds with a **GetAlertsSoapOut** response message.

The **GetAlertsSoapOut** response message contains common information and a list of information about each alert subscription.

3.1.4.2.1 Messages

3.1.4.2.1.1 GetAlertsSoapIn

The message represents the protocol client request for the current **alert subscriptions** from the protocol server.

The **SOAP action** value of the message is defined as:

```
http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts/GetAlerts
```

The **SOAP body** contains a **GetAlerts** element.

3.1.4.2.1.2 GetAlertsSoapOut

The message represents the protocol server response for a protocol client request for the list of current **alert subscriptions**.

The **SOAP body** contains a **GetAlertsResponse** element.

3.1.4.2.2 Elements

3.1.4.2.2.1 GetAlerts

GetAlerts is an empty structure indicating that a **GetAlerts WSDL** operation is being requested.

```
<s:element name="GetAlerts">  
  <s:complexType/>  
</s:element>
```

3.1.4.2.2.2 GetAlertsResponse

This structure contains the response to a **GetAlerts WSDL** operation.

```
<s:element name="GetAlertsResponse">  
  <s:complexType>  
    <s:sequence>  
      <s:element name="GetAlertsResult" type="tns:AlertInfoDefinition"/>  
    </s:sequence>  
  </s:complexType>  
</s:element>
```

GetAlertsResult: A structure of type **AlertInfoDefinition**, as specified in **AlertInfoDefinition** (section [3.1.4.2.3.1](#)).

3.1.4.2.3 Complex Types

3.1.4.2.3.1 AlertInfoDefinition

The **AlertInfoDefinition** structure contains common information for the **authenticated user** listed as **current user**, in addition to a list of alert subscriptions.

```
<s:complexType name="AlertInfoDefinition">
  <s:sequence>
    <s:element name="CurrentUser" type="s:string" minOccurs="0"/>
    <s:element name="AlertServerName" type="s:string" minOccurs="0"/>
    <s:element name="AlertServerUrl" type="s:string" minOccurs="0"/>
    <s:element name="AlertServerType" type="s:string" minOccurs="0"/>
    <s:element name="AlertsManagementUrl" type="s:string" minOccurs="0"/>
    <s:element name="AlertWebTitle" type="s:string" minOccurs="0"/>
    <s:element name="NewAlertUrl" type="s:string" minOccurs="0"/>
    <s:element name="AlertWebId" type="s:string" minOccurs="0"/>
    <s:element name="Alerts" type="tns:ArrayOfAlertDefinition"
      minOccurs="0"/>
  </s:sequence>
</s:complexType>
```

CurrentUser: The display name of the authenticated user for the current connection.

AlertServerName: The name of the server storing the current list of **alert subscriptions**. This name MUST conform to "Domain Names- Implementation and Specification", [\[RFC1035\]](#) section 2.3.1, and "Uniform Resource Identifiers (URI): Generic Syntax", [\[RFC2396\]](#). This is the host component of the request **URL**.

AlertServerUrl: A **URI** for the **site** storing the alert subscriptions. This MUST be the site URL.

AlertServerType: A string identifying the server software storing the alert subscriptions. [<3>](#)

AlertsManagementUrl: A URL to a page for the current list of alert subscriptions. This MUST reference a Web page that an authenticated user can request that summarizes the user's current alert subscriptions and allows for their management.

AlertWebTitle: The title of the site on which the alert subscriptions are stored.

NewAlertUrl: A URL that represents a site page that allows an authenticated user to create new alert subscriptions.

AlertWebId: The identifier of the site on which the alert subscriptions are stored. This MUST contain a string representing a GUID that uniquely identifies the site containing the returned alert subscriptions.

Alerts: The alert subscriptions. This represents a list of alert subscription structures that contain information about each current alert subscription.

3.1.4.2.3.2 ArrayOfAlertDefinition

This structure contains a list of zero or more alert subscription structures.

```
<s:complexType name="ArrayOfAlertDefinition">
  <s:sequence>
    <s:element name="Alert" type="tns:Alert"
      minOccurs="0" maxOccurs="unbounded"/>
  </s:sequence>
</s:complexType>
```

Alert: A single **Alert** subscription structure, as specified in section [3.1.4.2.3.3](#).

3.1.4.2.3.3 Alert

This structure represents a single **alert subscription**.

```
<s:complexType name="Alert">
  <s:sequence>
    <s:element name="Id" type="s:string" minOccurs="0" maxOccurs="1"/>
    <s:element name="Title" type="s:string" minOccurs="0" maxOccurs="1"/>
    <s:element name="Active" type="s:boolean" minOccurs="1" maxOccurs="1"/>
    <s:element name="EventType" type="s:string" minOccurs="0" maxOccurs="1"/>
    <s:element name="AlertForTitle" type="s:string" minOccurs="0" maxOccurs="1"/>
    <s:element name="AlertForUrl" type="s:string" minOccurs="0" maxOccurs="1"/>
    <s:element name="EditAlertUrl" type="s:string" minOccurs="0" maxOccurs="1"/>
    <s:element name="DeliveryChannels"
      type="tns:ArrayOfDeliveryChannelDefinition" minOccurs="0" maxOccurs="1"/>
  </s:sequence>
</s:complexType>
```

Id: The identifier of an alert subscription. This MUST contain a string representing a **GUID** which uniquely identifies the alert subscription.

Title: A string specifying the friendly name of the alert subscription.

Active: A BOOLEAN value indicating whether this alert subscription will generate alerts. This MUST contain the value "true".

EventType: A string indicating the type of event that caused the alert subscription to generate a notification. This MUST be a value from the following table.

Value	Description
Add	Triggered when a new item is added.
Modify	Triggered when an item is modified.
Delete	Triggered when an item is deleted.
Discussion	Triggered when a Web discussion comment associated with an item is added, modified, or deleted.
All	Triggered when any of the previous values is triggered.

AlertForTitle: A string specifying the title of the resource, such as a **list** or a **list item** being monitored by the alert subscription.

AlertForUrl: The URL of the resource, such as a list or a list item being monitored. This MUST be an absolute URL.

EditAlertUrl: The **URL** referencing a Web page to be used to edit the alert subscription. This MUST be an absolute URL.

DeliveryChannels: A list of the methods used to provide notifications when an alert subscription is triggered.

3.1.4.2.3.4 ArrayOfDeliveryChannelDefinition

This structure contains a list of zero or more **DeliveryChannel** structures.

```
<s:complexType name="ArrayOfDeliveryChannelDefinition">
  <s:sequence>
    <s:element name="DeliveryChannel" type="tns:DeliveryChannel"

```

```
        nillable="true" minOccurs="0" maxOccurs="unbounded"/>
    </s:sequence>
</s:complexType>
```

DeliveryChannel: See **DeliveryChannel** (section [3.1.4.2.3.5](#)).

3.1.4.2.3.5 DeliveryChannel

A structure describing how a notification is to be delivered.

```
<s:complexType name="DeliveryChannel" abstract="true"/>
```

3.1.4.2.3.6 EmailChannel

This structure is used to provide the information required to deliver a notification through an electronic mail service.

```
<s:complexType name="EmailChannel">
  <s:complexContent mixed="false">
    <s:extension base="tns:DeliveryChannel">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="Frequency" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="Address" type="s:string" />
      </s:sequence>
    </s:extension>
  </s:complexContent>
</s:complexType>
```

Frequency: A string indicating how often the notification is sent out. The value **MUST** be listed in the following table.

Value	Description
Immediate	A notification is sent immediately.
Daily	A notification is sent during daily processing.
Weekly	A notification is sent during weekly processing.

Address: This is the destination for notifications. It is an Internet Message Format-compliant [\[RFC2822\]](#) string to be used in the recipient field of a message header, as specified in [\[RFC2822\]](#).

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

4 Protocol Examples

The overall scenario is that a protocol client requires the list of **alert subscriptions** for the current **authenticated user**. The client submits a **GetAlerts** request similar to the following code example.

```
<?xml version="1.0"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope">
  <Body>
    <GetAlerts
      xmlns="http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts">
    </GetAlerts>
  </Body>
</Envelope>
```

The protocol server receives this request and creates an appropriate response, similar to the following code example.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <GetAlertsResponse
      xmlns="http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts">
      <GetAlertsResult>
        <CurrentUser>Auricchio, Jose Luis</CurrentUser>
        <AlertServerName>widgets</AlertServerName>
        <AlertServerUrl>http://widgets</AlertServerUrl>
        <AlertServerType>STS</AlertServerType>
        <AlertsManagementUrl>
          http://widgets/ layouts/MySubs.aspx
        </AlertsManagementUrl>
        <AlertWebTitle>widgets</AlertWebTitle>
        <NewAlertUrl>http://widgets/_layouts/SubChoos.aspx</NewAlertUrl>
        <AlertWebId>4a3d9478-00a0-4ea8-bdb6-36034a189433</AlertWebId>
        <Alerts>
          <Alert>
            <Id>{76061063-9C09-4C4D-B1A1-16D3F0CDF1F8}</Id>
            <Title>Shared Documents</Title>
            <Active>true</Active>
            <EventType>All</EventType>
            <AlertForUrl>http://widgets/Shared Documents</AlertForUrl>
            <EditAlertUrl>http://widgets/_layouts/SubEdit.aspx?Alert={76061063-9C09-4C4D-
              B1A1-16D3F0CDF1F8}&List={071C725D-EF5E-4779-B95C-0F8173C260E1}</EditAlertUrl>
            <DeliveryChannels>
              <DeliveryChannel xsi:type="EmailChannel">
                <Frequency>Immediate</Frequency>
                <Address>jose@cohowinery.com</Address>
              </DeliveryChannel>
            </DeliveryChannels>
          </Alert>
        </Alerts>
      </GetAlertsResult>
    </GetAlertsResponse>
  </soap:Body>
</soap:Envelope>
```

If the user directs the protocol client to delete an existing alert subscription, the request would resemble the following code example.

```
<?xml version="1.0"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope">
  <Body>
```

```
<DeleteAlerts
xmlns="http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts">
  <IDs>
    <string>{76061063-9C09-4C4D-B1A1-16D3F0CDF1F8}</string>
  </IDs>
</DeleteAlerts>
</Body>
</Envelope>
```

The protocol server response would typically resemble the following code example.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <DeleteAlertsResponse
xmlns="http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts">
      <DeleteAlertsResult />
    </DeleteAlertsResponse>
  </soap:Body>
</soap:Envelope>
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Full WSDL

For ease of implementation, the full **WSDL** is provided here.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
targetNamespace="http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts/">
      <s:element name="GetAlerts">
        <s:complexType />
      </s:element>
      <s:element name="GetAlertsResponse">
        <s:complexType>
          <s:sequence>
            <s:element name="GetAlertsResult" type="tns:AlertInfoDefinition"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="AlertInfoDefinition">
        <s:sequence>
          <s:element name="CurrentUser" type="s:string" minOccurs="0"/>
          <s:element name="AlertServerName" type="s:string" minOccurs="0"/>
          <s:element name="AlertServerUrl" type="s:string" minOccurs="0"/>
          <s:element name="AlertServerType" type="s:string" minOccurs="0"/>
          <s:element name="AlertsManagementUrl" type="s:string" minOccurs="0"/>
          <s:element name="AlertWebTitle" type="s:string" minOccurs="0"/>
          <s:element name="NewAlertUrl" type="s:string" minOccurs="0"/>
          <s:element name="AlertWebId" type="s:string" minOccurs="0"/>
          <s:element name="Alerts" type="tns:ArrayOfAlertDefinition" minOccurs="0"/>
        </s:sequence></s:complexType>
      <s:complexType name="ArrayOfAlertDefinition">
        <s:sequence>
          <s:element name="Alert" type="tns:Alert" minOccurs="0" maxOccurs="unbounded"/>
        </s:sequence></s:complexType>
      <s:complexType name="Alert">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1"
name="Id" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1"
name="Title" type="s:string" />
          <s:element minOccurs="1" maxOccurs="1"
name="Active" type="s:boolean" />
          <s:element minOccurs="0" maxOccurs="1"
name="EventType" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1"
name="AlertForTitle" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1"
name="AlertForUrl" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1"
name="EditAlertUrl" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1"
name="DeliveryChannels"
type="tns:ArrayOfDeliveryChannelDefinition" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="ArrayOfDeliveryChannelDefinition">
        <s:sequence>
          <s:element name="DeliveryChannel" type="tns:DeliveryChannel"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
        </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>

```



```

<s:complexType name="DeliveryChannel" abstract="true" />
<s:complexType name="EmailChannel">
  <s:complexContent mixed="false">
    <s:extension base="tns:DeliveryChannel">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
          name="Frequency" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
          name="Address" type="s:string" />
      </s:sequence>
    </s:extension>
  </s:complexContent>
</s:complexType>
<s:element name="DeleteAlerts">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
        name="IDs" type="tns:ArrayOfString" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:complexType name="ArrayOfString">
  <s:sequence>
    <s:element name="string" type="s:string" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
  </s:sequence>
</s:complexType>
<s:element name="DeleteAlertsResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1"
        name="DeleteAlertsResult"
        type="tns:ArrayOfDeleteFailureDefinition" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:complexType name="ArrayOfDeleteFailureDefinition">
  <s:sequence>
    <s:element name="DeleteFailure"
      type="tns>DeleteFailureDefinition" minOccurs="0"
maxOccurs="unbounded"/>
  </s:sequence>
</s:complexType>
<s:complexType name="DeleteFailureDefinition">
  <s:sequence>
    <s:element name="ID" type="s:string" minOccurs="0"/>
    <s:element name="Error" type="tns:ErrorType"/>
  </s:sequence>
</s:complexType>
<s:simpleType name="ErrorType">
  <s:restriction base="s:string">
    <s:enumeration value="None" />
    <s:enumeration value="AccessDenied" />
    <s:enumeration value="ServerError" />
    <s:enumeration value="TooManyErrors" />
  </s:restriction>
</s:simpleType>
</s:schema>
</wsdl:types>
<wsdl:message name="GetAlertsSoapIn">
  <wsdl:part name="parameters" element="tns:GetAlerts" />
</wsdl:message>
<wsdl:message name="GetAlertsSoapOut">
  <wsdl:part name="parameters" element="tns:GetAlertsResponse" />
</wsdl:message>
<wsdl:message name="DeleteAlertsSoapIn">
  <wsdl:part name="parameters" element="tns>DeleteAlerts" />
</wsdl:message>
<wsdl:message name="DeleteAlertsSoapOut">

```

```

    <wsdl:part name="parameters" element="tns:DeleteAlertsResponse" />
  </wsdl:message>
  <wsdl:portType name="AlertsSoap">
    <wsdl:operation name="GetAlerts">
      <wsdl:input message="tns:GetAlertsSoapIn" />
      <wsdl:output message="tns:GetAlertsSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="DeleteAlerts">
      <wsdl:input message="tns:DeleteAlertsSoapIn" />
      <wsdl:output message="tns:DeleteAlertsSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="AlertsSoap" type="tns:AlertsSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetAlerts">
      <soap:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts/GetAlerts"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DeleteAlerts">
      <soap:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts/DeleteAlerts"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="AlertsSoap12" type="tns:AlertsSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetAlerts">
      <soap12:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts/GetAlerts"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DeleteAlerts">
      <soap12:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/2002/1/alerts/DeleteAlerts"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>

```

7 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

- Microsoft Office Outlook 2007
- Microsoft SharePoint Foundation 2010
- Windows SharePoint Services 2.0
- Windows SharePoint Services 3.0
- Microsoft SharePoint Foundation 2013
- Microsoft SharePoint Server 2016
- Microsoft SharePoint Server 2019
- Microsoft SharePoint Server Subscription Edition

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

[<1> Section 3.1.4.1](#): Windows SharePoint Services 2.0, Windows SharePoint Services 3.0, SharePoint Foundation 2010, and SharePoint Foundation 2013 limit errors in the processing of **DeleteAlerts** to 20 errors in all released versions. The protocol client and protocol server are not dependent on this particular behavior.

[<2> Section 3.1.4.1.4.1](#): Windows SharePoint Services 2.0, Windows SharePoint Services 3.0, SharePoint Foundation 2010, and SharePoint Foundation 2013 never return this value.

[<3> Section 3.1.4.2.3.1](#): Windows SharePoint Services 2.0, Windows SharePoint Services 3.0, SharePoint Foundation 2010, and SharePoint Foundation 2013 set this value to "STS".

8 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
Z Appendix B: Product Behavior	Updated list of supported products.	Major

9 Index

A

Abstract data model
[server](#) 13
[Alert deletion](#) 9
[Alert enumeration](#) 9
[Applicability](#) 10
[Attribute groups](#) 12
[Attributes](#) 12

C

[Capability negotiation](#) 10
[Change tracking](#) 28
Client
[overview](#) 13
[Complex types](#) 11

D

Data model - abstract
[server](#) 13
Deletion
[alert](#) 9

E

Enumeration
[alert](#) 9
Events
[local - server](#) 20
[timer - server](#) 20
Examples
[overview](#) 21

F

[Fields - vendor-extensible](#) 10
[Full WSDL](#) 24

G

[Glossary](#) 6
[Groups](#) 12

I

[Implementer - security considerations](#) 23
[Index of security parameters](#) 23
[Informative references](#) 8
Initialization
[server](#) 13
[Introduction](#) 6

L

Local events
[server](#) 20

M

Message processing
[server](#) 13
Messages
[attribute groups](#) 12
[attributes](#) 12
[complex types](#) 11
[elements](#) 11
[enumerated](#) 11
[groups](#) 12
[namespaces](#) 11
[simple types](#) 12
[syntax](#) 11
[transport](#) 11

N

[Namespaces](#) 11
[Normative references](#) 8

O

Operations
[DeleteAlerts](#) 14
[GetAlerts](#) 16
[Overview \(synopsis\)](#) 8

P

[Parameters - security index](#) 23
[Preconditions](#) 10
[Prerequisites](#) 10
[Product behavior](#) 27
Protocol Details
[overview](#) 13

R

[References](#) 7
[informative](#) 8
[normative](#) 8
[Relationship to other protocols](#) 9

S

Security
[implementer considerations](#) 23
[parameter index](#) 23
Sequencing rules
[server](#) 13
Server
[abstract data model](#) 13
[DeleteAlerts operation](#) 14
[GetAlerts operation](#) 16
[initialization](#) 13
[local events](#) 20
[message processing](#) 13
[overview](#) 13
[sequencing rules](#) 13
[timer events](#) 20
[timers](#) 13

[Simple types](#) 12
[Standards assignments](#) 10
Syntax
 [messages - overview](#) 11

T

Timer events
 [server](#) 20
Timers
 [server](#) 13
[Tracking changes](#) 28
[Transport](#) 11
Types
 [complex](#) 11
 [simple](#) 12

V

[Vendor-extensible fields](#) 10
[Versioning](#) 10

W

[WSDL](#) 24