

[MS-SSDPWP]:

Database Publishing Wizard 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
8/7/2009	0.1	Major	First release.
11/6/2009	0.1.1	Editorial	Changed language and formatting in the technical content.
3/5/2010	0.2	Minor	Clarified the meaning of the technical content.
4/21/2010	0.3	Minor	Clarified the meaning of the technical content.
6/4/2010	0.3.1	Editorial	Changed language and formatting in the technical content.
9/3/2010	0.3.1	None	No changes to the meaning, language, or formatting of the technical content.
2/9/2011	0.3.1	None	No changes to the meaning, language, or formatting of the technical content.
7/7/2011	0.3.1	None	No changes to the meaning, language, or formatting of the technical content.
11/3/2011	0.3.1	None	No changes to the meaning, language, or formatting of the technical content.
1/19/2012	0.3.1	None	No changes to the meaning, language, or formatting of the technical content.
2/23/2012	4.0	Major	Updated and revised the technical content.
3/27/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
5/24/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
6/29/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	5.0	Major	Updated and revised the technical content.
10/23/2012	5.0	None	No changes to the meaning, language, or formatting of the technical content.
3/26/2013	5.0	None	No changes to the meaning, language, or formatting of the technical content.
6/11/2013	5.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	5.0	None	No changes to the meaning, language, or formatting of the technical content.
12/5/2013	5.0	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2014	6.0	Major	Updated and revised the technical content.
5/20/2014	6.0	None	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
5/10/2016	7.0	Major	Significantly changed the technical content.
8/16/2017	8.0	Major	Significantly changed the technical content.
10/16/2019	9.0	Major	Significantly changed the technical content.

Table of Contents

1	Introduction	6
1.1	Glossary	6
1.2	References	6
1.2.1	Normative References	6
1.2.2	Informative References	7
1.3	Overview	7
1.4	Relationship to Other Protocols	7
1.5	Prerequisites/Preconditions	8
1.6	Applicability Statement	8
1.7	Versioning and Capability Negotiation	8
1.8	Vendor-Extensible Fields	8
1.9	Standards Assignments.....	9
2	Messages.....	10
2.1	Transport	10
2.2	Common Message Syntax	10
2.2.1	Namespaces	10
2.2.2	Messages.....	10
2.2.3	Elements	10
2.2.4	Complex Types.....	10
2.2.5	Simple Types	10
2.2.6	Attributes	10
2.2.7	Groups	11
2.2.8	Attribute Groups.....	11
2.3	Directory Service Schema Elements	11
3	Protocol Details	12
3.1	PublishServiceSoap Server Details.....	12
3.1.1	Abstract Data Model.....	12
3.1.2	Timers	12
3.1.3	Initialization.....	12
3.1.4	Message Processing Events and Sequencing Rules	12
3.1.4.1	BeginPublish.....	13
3.1.4.1.1	Messages	13
3.1.4.1.1.1	BeginPublishSoapIn.....	14
3.1.4.1.1.2	BeginPublishSoapOut	14
3.1.4.1.2	Elements.....	14
3.1.4.1.2.1	BeginPublish	14
3.1.4.1.2.2	BeginPublishResponse	14
3.1.4.2	CancelPublish	15
3.1.4.2.1	Messages	15
3.1.4.2.1.1	CancelPublishSoapIn	15
3.1.4.2.1.2	CancelPublishSoapOut	15
3.1.4.2.2	Elements.....	15
3.1.4.2.2.1	CancelPublish	15
3.1.4.2.2.2	CancelPublishResponse.....	16
3.1.4.3	EndPublish	16
3.1.4.3.1	Messages	16
3.1.4.3.1.1	EndPublishSoapIn	16
3.1.4.3.1.2	EndPublishSoapOut	16
3.1.4.3.2	Elements.....	17
3.1.4.3.2.1	EndPublish	17
3.1.4.3.2.2	EndPublishResponse.....	17
3.1.4.4	GetServiceOptions	17
3.1.4.4.1	Messages	17

3.1.4.4.1.1	GetServiceOptionsSoapIn	17
3.1.4.4.1.2	GetServiceOptionsSoapOut	18
3.1.4.4.2	Elements	18
3.1.4.4.2.1	GetServiceOptions	18
3.1.4.4.2.2	GetServiceOptionsResponse	18
3.1.4.5	PublishData	18
3.1.4.5.1	Messages	19
3.1.4.5.1.1	PublishDataSoapIn	19
3.1.4.5.1.2	PublishDataSoapOut	19
3.1.4.5.2	Elements	19
3.1.4.5.2.1	PublishData	19
3.1.4.5.2.2	PublishDataResponse	19
3.1.4.6	PublishScript	20
3.1.4.6.1	Messages	20
3.1.4.6.1.1	PublishScriptSoapIn	20
3.1.4.6.1.2	PublishScriptSoapOut	20
3.1.4.6.2	Elements	20
3.1.4.6.2.1	PublishScript	20
3.1.4.6.2.2	PublishScriptResponse	21
3.1.5	Timer Events	21
3.1.6	Other Local Events	21
4	Protocol Examples	22
5	Security	24
5.1	Security Considerations for Implementers	24
5.2	Index of Security Parameters	24
6	Appendix A: Full WSDL	25
7	Appendix B: Product Behavior	30
8	Change Tracking	31
9	Index	32

1 Introduction

The Database Publishing Wizard Protocol is a format that enables communication with an instance of Microsoft SQL Server by using open, industry-standard protocols. By using this format, a publishing session can be initiated, data can be published, and scripts can be executed against an instance of SQL Server.

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:

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 message: An abstract, typed definition of the data that is communicated during a WSDL operation [\[WSDL\]](#). Also, an element that describes the data being exchanged between web service providers and clients.

XML: The Extensible Markup Language, as described in [\[XML1.0\]](#).

XML document: A document object that is well formed, as described in [\[XML10/5\]](#), and might be valid. An XML document has a logical structure that is composed of declarations, elements, comments, character references, and processing instructions. It also has a physical structure that is composed of entities, starting with the root, or document, entity.

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.

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

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

[SOAP1.2-1/2003] 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>

[WSDLSOAP] Angelov, D., Ballinger, K., Butek, R., et al., "WSDL 1.1 Binding Extension for SOAP 1.2", W3C Member Submission, April 2006, <http://www.w3.org/Submission/2006/SUBM-wsdl11soap12-20060405/>

[WSDL] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1", W3C Note, March 2001, <https://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, <https://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, <https://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, <https://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>

1.2.2 Informative References

[DPS] Microsoft Corporation, "SQL Server Hosting Web Service (and toolkit): Database Publishing Services", <http://sqlhost.codeplex.com/wikipage?title=Database%20Publishing%20Services&referringTitle=Home>

[MSDN-CRED] Microsoft Corporation, "Credentials (Database Engine)", <https://learn.microsoft.com/en-us/sql/relational-databases/security/authentication-access/credentials-database-engine>

[MSDN-Transact-SQLRef] Microsoft Corporation, "Transact-SQL Reference (Database Engine)", <https://learn.microsoft.com/en-us/sql/t-sql/language-reference>

1.3 Overview

The Database Publishing Wizard Protocol enables a user to publish an existing database to a remote server via a web service. This enables database deployment in hosted scenarios without requiring direct access to the database server.

1.4 Relationship to Other Protocols

The Database Publishing Wizard Protocol uses SOAP over HTTP or HTTPS as shown in the following layering diagram.



Figure 1: SOAP over HTTP or HTTPS

1.5 Prerequisites/Preconditions

Before using the Database Publishing Wizard Protocol, it is necessary to install and configure an instance of the Database Publishing Services web service. For more information about how to do this, see [\[DPS\]](#).

1.6 Applicability Statement

The Database Publishing Wizard Protocol is applicable whenever a user wants to deploy a database but the target instance of SQL Server is not accessible by using the typical SQL Server client tools, such as SQL Server Management Studio, SQL Server Management Objects (SMO), or Microsoft ADO.NET. The Database Publishing Wizard Protocol allows the deployment to occur via a proxy web service that does have direct access to the target instance of SQL Server.

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](#).
- **Protocol Versions:** This protocol has a separate **Web Services Description Language (WSDL)** port type for each version of the protocol. The operations that are available through each port are identical. The two ports support clients using SOAP 1.1 and SOAP 1.2, respectively.
- **Localization:** This protocol includes text strings in various messages. Localization considerations for such strings are specified in sections [2.2](#) and [3.1.4](#).
- **Capability Negotiation:** This protocol does explicit negotiation.

There is currently only one version of the protocol (version 1.1). However, clients can confirm that they are communicating with version 1.1 by invoking the [GetServiceOptions](#) method and then checking the **service_version** return value.

1.8 Vendor-Extensible Fields

This protocol does not include any vendor-extensible fields.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The web service message protocols SOAP 1.1, as specified in [\[SOAP1.1\]](#), and SOAP 1.2, as specified in [\[SOAP1.2-1/2003\]](#), are supported.

2.2 Common Message Syntax

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

2.2.1 Namespaces

This specification defines and references various **XML namespaces** by using the mechanisms that are specified in [\[XMLNS\]](#), as listed in the following table. 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/sqlserver/2006/12/publishing	Appendix A
s	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1]
soap12	http://schemas.xmlsoap.org/wsdl/soap12/	[WSDLSOAP]
targetNamespace	http://schemas.microsoft.com/sqlserver/2006/12/publishing	Appendix A
wsdl	http://schemas.xmlsoap.org/wsdl	[WSDL]

2.2.2 Messages

This specification does not contain any common **WSDL messages**.

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.

2.3 Directory Service Schema Elements

None.

3 Protocol Details

The client side of this protocol is simply a pass-through. That is, 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.

3.1 PublishServiceSoap Server Details

This section describes the server behavior of the Database Publishing Wizard Protocol. This port type supports the following **WSDL** operations:

- [BeginPublish](#)
- [CancelPublish](#)
- [EndPublish](#)
- [GetServiceOptions](#)
- [PublishData](#)
- [PublishScript](#)

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 that described in this document.

In the following sections:

- "State" refers to the condition that the database publishing wizard transitions into, in response to a request from the client. The [MS-SSDPWP] protocol itself is stateless.
- "Argument" refers to parameters and attributes that qualify an operation requested by the client.
- If the server responds with an error to the client while it is in a particular state, the server continues to remain in the same state without transitioning to a new state.
- The server's initial state is "Not Publishing".

3.1.2 Timers

None.

3.1.3 Initialization

The Database Publishing Wizard Protocol is initialized by invoking the [BeginPublish](#) operation with valid arguments to initiate a connection to an instance of SQL Server. When the **BeginPublish** operation is invoked, the server MUST connect to the specified instance of SQL Server to validate the arguments.

3.1.4 Message Processing Events and Sequencing Rules

The following table summarizes the list of **WSDL** operations as defined by this specification.

Operation	Description
BeginPublish	Sets the state to "Publish".
CancelPublish	Sets the state to "Not Publishing".
EndPublish	Sets the state to "Not Publishing".
GetServiceOptions	Returns an XML document that has pertinent hosting-provider-specific configuration values and other account metadata.
PublishData	Saves data to database tables. Can be invoked only if the state of the service is "Publish" (set by calling BeginPublish).
PublishScript	Executes the passed string as a Transact-SQL script against the database. Can be invoked only if the state of the service is "Publish" (set by calling BeginPublish).

3.1.4.1 BeginPublish

```
<wsdl:operation name="BeginPublish">
  <wsdl:input message="tns:BeginPublishSoapIn" />
  <wsdl:output message="tns:BeginPublishSoapOut" />
</wsdl:operation>
```

The **BeginPublish** operation is used to transition the server into the "Publish" state. The connection information and user credentials that are necessary to open a connection to an instance of the server are passed in by using this operation. The **useTransactions** argument determines whether subsequent operations are performed transactionally.

BeginPublish MUST initiate a connection to the specified instance of the server by using the provided arguments.

The specific error returned to the client can vary as follows.

In the case of missing (empty string) arguments for the *serverName*, *databaseName*, *sqlUsername*, or *sqlPassword* parameters, an error in the following form is returned:

```
System.ArgumentException: Null values not allowed for parameters for BeginPublish.
```

In the case of invalid database credentials (a non-existent server, database, or username, or an invalid password), an error is returned. The error returned is in the following form:

```
Microsoft.SqlServer.Hosting.Service.SqlErrorException: An error occurred while trying to begin publish.
```

Invoking **BeginPublish** when not in the "Not Publishing" state results in an error being returned to the client. The error returned is in the following form:

```
Microsoft.SqlServer.Hosting.Service.PublishOccurringException: There is a current valid publishing session occurring. You must close it before you can begin a new one.
```

3.1.4.1.1 Messages

The following **WSDL message** definitions are specific to this operation.

3.1.4.1.1.1 BeginPublishSoapIn

The **BeginPublishSoapIn WSDL message** has one parameter, *BeginPublish*. The [BeginPublish](#) element contains connection information for the instance of SQL Server and user authentication information.

```
<wsdl:message name="BeginPublishSoapIn">
  <wsdl:part name="parameters" element="tns:BeginPublish" />
</wsdl:message>
```

3.1.4.1.1.2 BeginPublishSoapOut

The **BeginPublishSoapOut WSDL message** has one parameter, *BeginPublishResponse*.

```
<wsdl:message name="BeginPublishSoapOut">
  <wsdl:part name="parameters" element="tns:BeginPublishResponse" />
</wsdl:message>
```

3.1.4.1.2 Elements

The following **XML schema** element definitions are specific to this operation.

3.1.4.1.2.1 BeginPublish

```
<s:element name="BeginPublish">
  <s:complexType>
    <s:sequence>
      <s:element name="serverName" type="s:string" minOccurs="0" maxOccurs="1" />
      <s:element name="databaseName" type="s:string" minOccurs="0" maxOccurs="1" />
      <s:element minOccurs="0" type="s:string" maxOccurs="1" name="sqlUsername" />
      <s:element name="sqlPassword" type="s:string" minOccurs="0" maxOccurs="1" />
      <s:element name="useTransactions" type="s:boolean" minOccurs="1" maxOccurs="1" />
    </s:sequence>
  </s:complexType>
</s:element>
```

The **BeginPublish** element represents the target instance of SQL Server, user credentials, and desired transactional behavior of subsequent operations.

The **serverName** element represents the name of the target instance of SQL Server.

The **databaseName** element represents the name of the target database on the **serverName** instance of SQL Server.

The **sqlUsername** element represents the user name that is used to authenticate on the target instance of SQL Server.

The **sqlPassword** element represents the password that is used to authenticate on the target instance of SQL Server.

The **useTransactions** element represents the desired transactional behavior of subsequent server operations.

For more information about the credentials that are used to connect to SQL Server, see [\[MSDN-CRED\]](#).

3.1.4.1.2.2 BeginPublishResponse

The following code is the XSD for the **BeginPublish** operation response.

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

3.1.4.2 CancelPublish

```
<wsdl:operation name="CancelPublish">
  <wsdl:input message="tns:CancelPublishSoapIn" />
  <wsdl:output message="tns:CancelPublishSoapOut" />
</wsdl:operation>
```

The **CancelPublish** operation is used to transition the server into the "Not Publishing" state. The connection to the instance of the server that was initialized by the [BeginPublish](#) operation is closed. In addition, if **BeginPublish** was invoked by using a value of true for the **useTransactions** argument, the transaction that covers all the changes made by the [PublishScript](#) and [PublishData](#) operations is rolled back.

Invoking **CancelPublish** when not in the "Publish" state results in an error being returned to the client. The error returned is in the following form:

```
Microsoft.SqlServer.Hosting.Service.NotPublishingException: A valid publishing session must
be started before it can be ended.
```

Invoking **CancelPublish** MUST release the connection to the target instance of the server.

3.1.4.2.1 Messages

The following **WSDL message** definitions are specific to this operation.

3.1.4.2.1.1 CancelPublishSoapIn

The **CancelPublishSoapIn WSDL message** has one parameter, *CancelPublish*.

```
<wsdl:message name="CancelPublishSoapIn">
  <wsdl:part name="parameters" element="tns:CancelPublish" />
</wsdl:message>
```

3.1.4.2.1.2 CancelPublishSoapOut

The **CancelPublishSoapOut WSDL message** has one parameter, *CancelPublishResponse*.

```
<wsdl:message name="CancelPublishSoapOut">
  <wsdl:part name="parameters" element="tns:CancelPublishResponse" />
</wsdl:message>
```

3.1.4.2.2 Elements

The following **XML schema** element definitions are specific to the **CancelPublish** operation.

3.1.4.2.2.1 CancelPublish

The following example shows the XSD for the **CancelPublish** operation request.

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

3.1.4.2.2 CancelPublishResponse

The following example shows the XSD for the **CancelPublish** operation response.

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

3.1.4.3 EndPublish

```
<wsdl:operation name="EndPublish">
  <wsdl:input message="tns:EndPublishSoapIn" />
  <wsdl:output message="tns:EndPublishSoapOut" />
</wsdl:operation>
```

The **EndPublish** operation is used to transition the server into the "Not Publishing" state. The connection to the instance of the server that was initialized by the [BeginPublish](#) operation is closed. In addition, if **BeginPublish** was invoked by using a value of True for the **useTransactions** argument, the transaction that covers all the changes made by the [PublishScript](#) and [PublishData](#) operations is committed.

Invoking **EndPublish** when not in the "Publish" state results in an error being returned to the client. The error returned is in the following form:

```
Microsoft.SqlServer.Hosting.Service.NotPublishingException: A valid publishing session must
be started before it can be ended.
```

Invoking **EndPublish** MUST release the connection to the target instance of the server.

3.1.4.3.1 Messages

The following **WSDL message** definitions are specific to this operation.

3.1.4.3.1.1 EndPublishSoapIn

The **EndPublishSoapIn WSDL message** has one parameter, *EndPublish*.

```
<wsdl:message name="EndPublishSoapIn">
  <wsdl:part name="parameters" element="tns:EndPublish" />
</wsdl:message>
```

3.1.4.3.1.2 EndPublishSoapOut

The **EndPublishSoapOut WSDL message** has one parameter, *EndPublishResponse*.

```
<wsdl:message name="EndPublishSoapOut">
  <wsdl:part name="parameters" element="tns:EndPublishResponse" />
```

```
</wsdl:message>
```

3.1.4.3.2 Elements

The following **XML schema** element definitions are specific to the [EndPublish](#) operation.

3.1.4.3.2.1 EndPublish

The following example shows the XSD for the **EndPublish** operation request.

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

3.1.4.3.2.2 EndPublishResponse

The following example shows the XSD for the **EndPublish** operation response.

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

3.1.4.4 GetServiceOptions

```
<wsdl:operation name="GetServiceOptions">  
  <wsdl:input message="tns:GetServiceOptionsSoapIn" />  
  <wsdl:output message="tns:GetServiceOptionsSoapOut" />  
</wsdl:operation>
```

The **GetServiceOptions** operation returns an **XML document** that has pertinent hosting-provider-specific configuration values and other account values.

As an example of the **XML** that is valid to return, the default implementation of the Database Publishing Services web service returns the following.

```
<options>  
  <max_request_length>4096</max_request_length>  
  <service_version>1.1.0.0</service_version>  
</options>
```

3.1.4.4.1 Messages

The following **WSDL message** definitions are specific to this operation.

3.1.4.4.1.1 GetServiceOptionsSoapIn

The **GetServiceOptionsSoapIn WSDL message** has one parameter, *GetServiceOptions*.

```
<wsdl:message name="GetServiceOptionsSoapIn">  
  <wsdl:part name="parameters" element="tns:GetServiceOptions" />  
</wsdl:message>
```

3.1.4.4.1.2 GetServiceOptionsSoapOut

The **GetServiceOptionsSoapOut WSDL message** has one parameter, *GetServiceOptionsResponse*.

```
<wsdl:message name="GetServiceOptionsSoapOut">
  <wsdl:part name="parameters" element="tns:GetServiceOptionsResponse" />
</wsdl:message>
```

3.1.4.4.2 Elements

The following **XML schema** element definitions are specific to this operation.

3.1.4.4.2.1 GetServiceOptions

The following example shows the XSD for the **GetServiceOptions** operation request.

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

3.1.4.4.2.2 GetServiceOptionsResponse

The following example shows the XSD for the **GetServiceOptions** operation response.

```
<s:element name="GetServiceOptionsResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="GetServiceOptionsResult" minOccurs="0" maxOccurs="1">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
```

The **GetServiceOptionsResult** element represents an **XML document** that contains pertinent hosting-provider-specific configuration values and other account metadata.

3.1.4.5 PublishData

The following example shows the XSD for the **GetServiceOptions** operation request.

```
<wsdl:operation name="PublishData">
  <wsdl:input message="tns:PublishDataSoapIn" />
  <wsdl:output message="tns:PublishDataSoapOut" />
</wsdl:operation>
```

The **PublishData** operation saves the state in a **DataSet** class to database tables. **PublishData** can be invoked only if the service is in the "Publish" state (set by calling [BeginPublish](#)).

In the case of invalid database credentials or inputs (nonexistent server, database, username, or invalid password), an error is returned.

3.1.4.5.1 Messages

The following **WSDL message** definitions are specific to this operation.

3.1.4.5.1.1 PublishDataSoapIn

The **PublishDataSoapIn WSDL message** has one parameter, *PublishData*.

```
<wsdl:message name="PublishDataSoapIn">
  <wsdl:part name="parameters" element="tns:PublishData" />
</wsdl:message>
```

3.1.4.5.1.2 PublishDataSoapOut

The **PublishDataSoapOut WSDL message** has one parameter, *PublishDataResponse*.

```
<wsdl:message name="PublishDataSoapOut">
  <wsdl:part name="parameters" element="tns:PublishDataResponse" />
</wsdl:message>
```

3.1.4.5.2 Elements

The following **XML schema** element definitions are specific to this operation.

3.1.4.5.2.1 PublishData

The following example shows the XSD for the **PublishData** operation request.

```
<s:element name="PublishData">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="ds">
        <s:complexType>
          <s:sequence>
            <s:element ref="s:schema" />
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
```

The **PublishData** element represents data that **MUST** be loaded into the target database.

The **ds** element represents an ADO.NET **DataSet** class serialized as **XML**.

3.1.4.5.2.2 PublishDataResponse

The following example shows the XSD for the **PublishData** operation response.

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

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

3.1.4.6 PublishScript

The following example shows the XSD for the **PublishData** operation request.

```
<wsdl:operation name="PublishScript">  
  <wsdl:input message="tns:PublishScriptSoapIn" />  
  <wsdl:output message="tns:PublishScriptSoapOut" />  
</wsdl:operation>
```

The **PublishScript** operation executes the passed-in string as a Transact-SQL script against the database. **PublishScript** can be invoked only if the service is in the "Publish" state (set by calling [BeginPublish](#)).

For more information about Transact-SQL, see [\[MSDN-Transact-SQLRef\]](#).

3.1.4.6.1 Messages

The following **WSDL message** definitions are specific to this operation.

3.1.4.6.1.1 PublishScriptSoapIn

The **PublishScriptSoapIn WSDL message** has one parameter, *PublishScript*.

```
<wsdl:message name="PublishScriptSoapIn">  
  <wsdl:part name="parameters" element="tns:PublishScript" />  
</wsdl:message>
```

3.1.4.6.1.2 PublishScriptSoapOut

The **PublishScriptSoapOut WSDL message** has one parameter, *PublishScriptResponse*.

```
<wsdl:message name="PublishScriptSoapOut">  
  <wsdl:part name="parameters" element="tns:PublishScriptResponse" />  
</wsdl:message>
```

3.1.4.6.2 Elements

The following **XML schema** element definitions are specific to this operation.

3.1.4.6.2.1 PublishScript

The following example shows the XSD for the **PublishScript** operation request.

```
<s:element name="PublishScript">  
  <s:complexType>  
    <s:sequence>  
      <s:element name="script" type="s:string" minOccurs="0" maxOccurs="1" />  
    </s:sequence>  
  </s:complexType>  
</s:element>
```

The **PublishScript** element represents a Transact-SQL script that MUST be executed against the target database.

3.1.4.6.2.2 PublishScriptResponse

The following code shows the XSD for the **PublishScript** operation response.

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

The **PublishScriptResponse** is a response to the **PublishScript** request containing the Transact-SQL script that is to be executed against a target database. The **PublishScriptSoapOut** WSDL message has one parameter, *PublishScriptResponse*.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

4 Protocol Examples

The following is an example of a procedure with which clients can use this protocol to create a simple database and then populate it with data.

First, the client begins by invoking the [BeginPublish](#) operation together with the valid server connection and user credentials, as shown in the following code example.

SOAP 1.1 BeginPublish Request

```
POST /Database%20Publishing%20Services%201.1/Publish/Service.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.microsoft.com/sqlserver/2006/12/publishing/BeginPublish"
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <BeginPublish
xmlns="http://schemas.microsoft.com/sqlserver/2006/12/publishing">
      <serverName>myServer</serverName>
      <databaseName>myDatabase</databaseName>
      <sqlUsername>myUsername</sqlUsername>
      <sqlPassword>myPassword</sqlPassword>
      <useTransactions>true</useTransactions>
    </BeginPublish>
  </soap:Body>
</soap:Envelope>
```

SOAP 1.1 BeginPublish Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <BeginPublishResponse
xmlns="http://schemas.microsoft.com/sqlserver/2006/12/publishing" />
  </soap:Body>
</soap:Envelope>
```

Then, the client invokes the [PublishScript](#) operation with a Transact-SQL script, which creates the new database and the database objects (for example, tables, views, and stored procedures), as shown in the following code example.

SOAP 1.1 PublishScript Request

```
POST /Database%20Publishing%20Services%201.1/Publish/Service.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.microsoft.com/sqlserver/2006/12/publishing/PublishScript"
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
```

```

    <PublishScript
xmlns="http://schemas.microsoft.com/sqlserver/2006/12/publishing">
    <script>create table table1(id int primary key)</script>
    </PublishScript>
</soap:Body>
</soap:Envelope>

```

SOAP 1.1 PublishScript Response

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <PublishScriptResponse
xmlns="http://schemas.microsoft.com/sqlserver/2006/12/publishing" />
    </soap:Body>
</soap:Envelope>

```

Lastly, the client invokes the [EndPublish](#) operation to finish the publishing session and to release server resources, as shown in the following code example.

SOAP 1.1 EndPublish Request

```

POST /Database%20Publishing%20Services%201.1/Publish/Service.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.microsoft.com/sqlserver/2006/12/publishing/EndPublish"
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <EndPublish
xmlns="http://schemas.microsoft.com/sqlserver/2006/12/publishing" />
    </soap:Body>
</soap:Envelope>

```

SOAP 1.1 EndPublish Response

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <EndPublishResponse
xmlns="http://schemas.microsoft.com/sqlserver/2006/12/publishing" />
    </soap:Body>
</soap:Envelope>

```

5 Security

5.1 Security Considerations for Implementers

The use of this protocol requires passing server identification information and user authentication credentials (user name and password) to the [BeginPublish](#) method. Furthermore, operations that are performed by the [PublishScript](#) and [PublishData](#) methods are executed by using the information that was previously supplied to **BeginPublish**. Therefore, it is important to help guarantee the security of the data transmission by using HTTPS or by securing the service behind a firewall that requires authentication.

5.2 Index of Security Parameters

None.

6 Appendix A: Full WSDL

For ease of implementation, this section provides the full **WSDL** for the Database Publishing Wizard Protocol.

```
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:tns="http://schemas.microsoft.com/sqlserver/2006/12/publishing"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  targetNamespace="http://schemas.microsoft.com/sqlserver/2006/12/publishing"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
      targetNamespace="http://schemas.microsoft.com/sqlserver/2006/12/publishing">
      <s:element name="BeginPublish">
        <s:complexType>
          <s:sequence>
            <s:element name="serverName" type="s:string" minOccurs="0" maxOccurs="1" />
            <s:element name="databaseName" type="s:string" minOccurs="0" maxOccurs="1" />
            <s:element name="sqlUsername" type="s:string" minOccurs="0" maxOccurs="1" />
            <s:element name="sqlPassword" type="s:string" minOccurs="0" maxOccurs="1" />
            <s:element name="useTransactions" type="s:boolean" minOccurs="1" maxOccurs="1" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="BeginPublishResponse">
        <s:complexType />
      </s:element>
      <s:element name="PublishScript">
        <s:complexType>
          <s:sequence>
            <s:element name="script" type="s:string" minOccurs="0" maxOccurs="1" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="PublishScriptResponse">
        <s:complexType />
      </s:element>
      <s:element name="PublishData">
        <s:complexType>
          <s:sequence>
            <s:element name="ds" minOccurs="0" maxOccurs="1" >
              <s:complexType>
                <s:sequence>
                  <s:element ref="s:schema" />
                  <s:any />
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="PublishDataResponse">
        <s:complexType />
      </s:element>
      <s:element name="EndPublish">
        <s:complexType />
      </s:element>
      <s:element name="EndPublishResponse">
        <s:complexType />
      </s:element>
      <s:element name="CancelPublish">
        <s:complexType />
      </s:element>
    </s:schema>
  </wsdl:types>

```

```

    </s:element>
    <s:element name="CancelPublishResponse">
      <s:complexType />
    </s:element>
    <s:element name="GetServiceOptions">
      <s:complexType />
    </s:element>
    <s:element name="GetServiceOptionsResponse">
      <s:complexType>
        <s:sequence>
          <s:element name="GetServiceOptionsResult" minOccurs="0" maxOccurs="1">
            <s:complexType mixed="true">
              <s:sequence>
                <s:any />
              </s:sequence>
            </s:complexType>
          </s:element>
        </s:sequence>
      </s:complexType>
    </s:element>
  </s:schema>
</wsdl:types>
<wsdl:message name="BeginPublishSoapIn">
  <wsdl:part name="parameters" element="tns:BeginPublish" />
</wsdl:message>
<wsdl:message name="BeginPublishSoapOut">
  <wsdl:part name="parameters" element="tns:BeginPublishResponse" />
</wsdl:message>
<wsdl:message name="PublishScriptSoapIn">
  <wsdl:part name="parameters" element="tns:PublishScript" />
</wsdl:message>
<wsdl:message name="PublishScriptSoapOut">
  <wsdl:part name="parameters" element="tns:PublishScriptResponse" />
</wsdl:message>
<wsdl:message name="PublishDataSoapIn">
  <wsdl:part name="parameters" element="tns:PublishData" />
</wsdl:message>
<wsdl:message name="PublishDataSoapOut">
  <wsdl:part name="parameters" element="tns:PublishDataResponse" />
</wsdl:message>
<wsdl:message name="EndPublishSoapIn">
  <wsdl:part name="parameters" element="tns:EndPublish" />
</wsdl:message>
<wsdl:message name="EndPublishSoapOut">
  <wsdl:part name="parameters" element="tns:EndPublishResponse" />
</wsdl:message>
<wsdl:message name="CancelPublishSoapIn">
  <wsdl:part name="parameters" element="tns:CancelPublish" />
</wsdl:message>
<wsdl:message name="CancelPublishSoapOut">
  <wsdl:part name="parameters" element="tns:CancelPublishResponse" />
</wsdl:message>
<wsdl:message name="GetServiceOptionsSoapIn">
  <wsdl:part name="parameters" element="tns:GetServiceOptions" />
</wsdl:message>
<wsdl:message name="GetServiceOptionsSoapOut">
  <wsdl:part name="parameters" element="tns:GetServiceOptionsResponse" />
</wsdl:message>
<wsdl:portType name="PublishServiceSoap">
  <wsdl:operation name="BeginPublish">
    <wsdl:input message="tns:BeginPublishSoapIn" />
    <wsdl:output message="tns:BeginPublishSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="PublishScript">
    <wsdl:input message="tns:PublishScriptSoapIn" />
    <wsdl:output message="tns:PublishScriptSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="PublishData">
    <wsdl:input message="tns:PublishDataSoapIn" />
  </wsdl:operation>

```

```

        <wsdl:output message="tns:PublishDataSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="EndPublish">
        <wsdl:input message="tns:EndPublishSoapIn" />
        <wsdl:output message="tns:EndPublishSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="CancelPublish">
        <wsdl:input message="tns:CancelPublishSoapIn" />
        <wsdl:output message="tns:CancelPublishSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="GetServiceOptions">
        <wsdl:input message="tns:GetServiceOptionsSoapIn" />
        <wsdl:output message="tns:GetServiceOptionsSoapOut" />
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="PublishServiceSoap" type="tns:PublishServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="BeginPublish">
        <soap:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/BeginPublish"
style="document" />
        <wsdl:input>
            <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="PublishScript">
        <soap:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/PublishScript"
style="document" />
        <wsdl:input>
            <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="PublishData">
        <soap:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/PublishData"
style="document" />
        <wsdl:input>
            <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="EndPublish">
        <soap:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/EndPublish"
style="document" />
        <wsdl:input>
            <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="CancelPublish">
        <soap:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/CancelPublish"
style="document" />
        <wsdl:input>
            <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>

```

```

        <soap:body use="literal" />
    </wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetServiceOptions">
    <soap:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/GetServiceOptions"
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="PublishServiceSoap12" type="tns:PublishServiceSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="BeginPublish">
        <soap12:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/BeginPublish"
style="document" />
        <wsdl:input>
            <soap12:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap12:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="PublishScript">
        <soap12:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/PublishScript"
style="document" />
        <wsdl:input>
            <soap12:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap12:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="PublishData">
        <soap12:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/PublishData"
style="document" />
        <wsdl:input>
            <soap12:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap12:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="EndPublish">
        <soap12:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/EndPublish"
style="document" />
        <wsdl:input>
            <soap12:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap12:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="CancelPublish">
        <soap12:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/CancelPublish"
style="document" />
        <wsdl:input>
            <soap12:body use="literal" />
        </wsdl:input>
        <wsdl:output>

```

```

        <soap12:body use="literal" />
    </wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetServiceOptions">
    <soap12:operation soapAction=
"http://schemas.microsoft.com/sqlserver/2006/12/publishing/GetServiceOptions"
style="document" />
    <wsdl:input>
        <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
        <soap12:body use="literal" />
    </wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="PublishService">
    <wsdl:port name="PublishServiceSoap" binding="tns:PublishServiceSoap">
        <soap:address location=
"http://localhost:44295/Database_Publsihing_Services_1.1/Publish/Service.asmx"
/>
    </wsdl:port>
    <wsdl:port name="PublishServiceSoap12" binding="tns:PublishServiceSoap12">
        <soap12:address location=
"http://localhost:44295/Database_Publsihing_Services_1.1/Publish/Service.asmx"
/>
    </wsdl:port>
</wsdl:service>
</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 SQL Server 2008
- Microsoft SQL Server 2008 R2
- Microsoft SQL Server 2012
- Microsoft SQL Server 2014
- Microsoft SQL Server 2016

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.

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	Removed SQL Server 2017 from the list of applicable products.	Major

9 Index

A

Abstract data model
[server](#) 12
[Applicability](#) 8
[Attribute groups](#) 11
[Attributes](#) 10

B

[BeginPublish element](#) 14
[BeginPublish operation](#) 13
[BeginPublishResponse element](#) 14
[BeginPublishSoapIn message](#) 14
[BeginPublishSoapOut message](#) 14

C

[CancelPublish element](#) 15
[CancelPublish operation](#) 15
[CancelPublishResponse element](#) 16
[CancelPublishSoapIn message](#) 15
[CancelPublishSoapOut message](#) 15
[Capability negotiation](#) 8
[Change tracking](#) 31
[Complex types](#) 10

D

Data model - abstract
[server](#) 12
[Directory service schema elements](#) 11

E

[Elements - directory service schema](#) 11
[EndPublish element](#) 17
[EndPublish operation](#) 16
[EndPublishResponse element](#) 17
[EndPublishSoapIn message](#) 16
[EndPublishSoapOut message](#) 16
Events
[local - server](#) 21
[timer - server](#) 21

F

[Fields - vendor-extensible](#) 8
[Full WSDL](#) 25

G

[GetServiceOptions element](#) 18
[GetServiceOptions operation](#) 17
[GetServiceOptionsResponse element](#) 18
[GetServiceOptionsSoapIn message](#) 17
[GetServiceOptionsSoapOut message](#) 18
[Glossary](#) 6
[Groups](#) 11

I

[Implementer - security considerations](#) 24
[Index of security parameters](#) 24
[Informative references](#) 7
Initialization
[server](#) 12
[Introduction](#) 6

L

Local events
[server](#) 21

M

Message processing
[server](#) 12
[Messages](#) 10
[attribute groups](#) 11
[attributes](#) 10
[complex types](#) 10
[elements](#) 10
[enumerated](#) 10
[groups](#) 11
[namespaces](#) 10
[simple types](#) 10
[syntax](#) 10
[transport](#) 10

N

[Namespaces](#) 10
[Normative references](#) 6

O

Operations
[BeginPublish](#) 13
[CancelPublish](#) 15
[EndPublish](#) 16
[GetServiceOptions](#) 17
[PublishData](#) 18
[PublishScript](#) 20
[Overview \(synopsis\)](#) 7

P

[Parameters - security index](#) 24
[Preconditions](#) 8
[Prerequisites](#) 8
[Product behavior](#) 30
Protocol Details
[overview](#) 12
[Protocol examples](#) 22
[PublishData element](#) 19
[PublishData operation](#) 18
[PublishDataResponse element](#) 19
[PublishDataSoapIn message](#) 19
[PublishDataSoapOut message](#) 19
[PublishScript element](#) 20
[PublishScript operation](#) 20

[PublishScriptResponse element](#) 21
[PublishScriptSoapIn message](#) 20
[PublishScriptSoapOut message](#) 20
[PublishServiceSoap server details](#) 12

R

[References](#) 6
 [informative](#) 7
 [normative](#) 6
[Relationship to other protocols](#) 7

S

[Schema elements - directory service](#) 11
[Security](#) 24
 [implementer considerations](#) 24
 [parameter index](#) 24
Sequencing rules
 [server](#) 12
Server
 [abstract data model](#) 12
 [BeginPublish operation](#) 13
 [CancelPublish operation](#) 15
 [EndPublish operation](#) 16
 [GetServiceOptions operation](#) 17
 [initialization](#) 12
 [local events](#) 21
 [message processing](#) 12
 [PublishData operation](#) 18
 [PublishScript operation](#) 20
 [sequencing rules](#) 12
 [timer events](#) 21
 [timers](#) 12
[Simple types](#) 10
[Standards assignments](#) 9
Syntax
 [messages - overview](#) 10

T

Timer events
 [server](#) 21
Timers
 [server](#) 12
[Tracking changes](#) 31
[Transport](#) 10
Types
 [complex](#) 10
 [simple](#) 10

V

[Vendor-extensible fields](#) 8
[Versioning](#) 8

W

[WSDL](#) 25