[MS-SSDPWP]:
Database Publishing Wizard Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
08/07/2009	0.1	Major	First release.
11/06/2009	0.1.1	Editorial	Revised and edited the technical content.
03/05/2010	0.2	Minor	Updated the technical content.
04/21/2010	0.3	Minor	Updated the technical content.
06/04/2010	0.3.1	Editorial	Revised and edited the technical content.
09/03/2010	0.3.1	No change	No changes to the meaning, language, or formatting of the technical content.
02/09/2011	0.3.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/07/2011	0.3.1	No change	No changes to the meaning, language, or formatting of the technical content.
11/03/2011	0.3.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/19/2012	0.3.1	No change	No changes to the meaning, language, or formatting of the technical content.
02/23/2012	4.0	Major	Significantly changed the technical content.

Contents

1 Introduction

This document specifies the Microsoft® SQL Server® Database Publishing Wizard Protocol [MS-SSDPWP]. This format allows for communication with an instance of 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.

1.1 Glossary

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

Web Services Description Language (WSDL)
WSDL message
XML
XML namespace
XML schema

The following terms are specific to this document:

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

1.2 References

References to Microsoft Open Specification documents do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

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

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

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

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

[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

[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/wsdl11soap12/

[XMLNS3] World Wide Web Consortium, "Namespaces in XML 1.0 (Third Edition)", December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

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

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

1.2.2 Informative References

[DPS] Microsoft Corporation, "SQL Server Hosting Toolkit: Database Publishing Services", http://sqlhost.codeplex.com/Wiki/View.aspx?title=Database%20Publishing%20Services&referringTitle=Home

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

[MSDN-CRED] Microsoft Corporation, "Credentials (Database Engine)", http://msdn.microsoft.com/en-us/library/ms161950.aspx.

[MSDN-TSQL-Ref] Microsoft Corporation, "Transact-SQL Reference", http://msdn.microsoft.com/en-us/library/ms189826(SQL.90).aspx

1.3 Protocol Overview (Synopsis)

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 as shown in the following layering diagram.

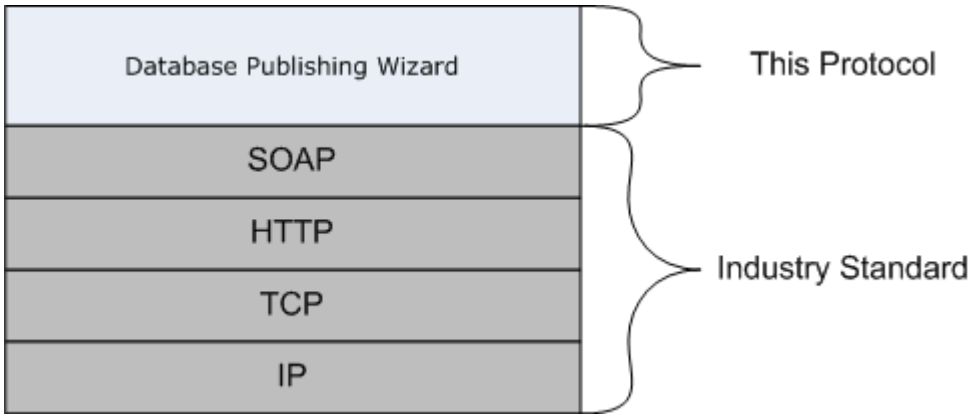

Figure 1: SOAP over HTTP

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

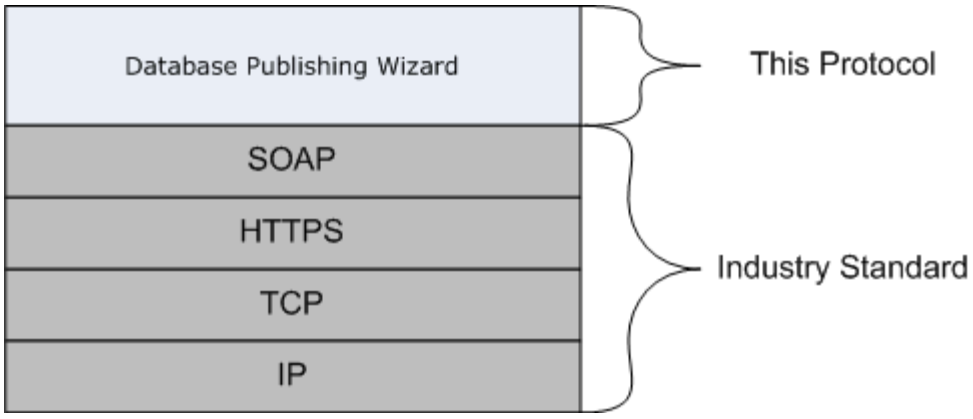

Figure 2: SOAP over HTTPS

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 Microsoft® 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 may 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 SOAP 1.1 Web service message protocol (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 [XMLNS3], 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 (non-existent server, database, username, or invalid password), an error is returned.<1>

Invoking **BeginPublish** when not in the "Not Publishing" state MUST result in an error being returned to the client.<2>

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 minOccurs="0" maxOccurs="1" name="serverName"
type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="databaseName"
type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="sqlUsername"
type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="sqlPassword"
type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="useTransactions"
type="s:boolean" />
          </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 BeginPublishResponse operation request.

```
<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 MUST result in an error being returned to the client.<3>

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>
```

3.1.4.2.2.2 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 MUST result in an error being returned to the client.<4>

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 request.

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

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-TSQL-Ref].

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 minOccurs="0" maxOccurs="1" name="script"
  type="s:string" />
            </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 must 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%20Publsihing%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%20Publsihing%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%20Publsihing%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 minOccurs="0" maxOccurs="1" name="serverName"
 type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="databaseName"
 type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="sqlUsername"
 type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="sqlPassword"
 type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="useTransactions"
 type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="BeginPublishResponse">
        <s:complexType />
      </s:element>
      <s:element name="PublishScript">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="script"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="PublishScriptResponse">
        <s:complexType />
      </s:element>
      <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>
      <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: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 minOccurs="0" maxOccurs="1"
name="GetServiceOptionsResult">
              <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: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 released service packs:

- Microsoft® SQL Server® 2008
- Microsoft® SQL Server® 2008 R2
- Microsoft® SQL Server® 2012

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product 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: The error returned is in the following form:

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

<2> Section 3.1.4.1: 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> Section 3.1.4.2: 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.
```

<4> Section 3.1.4.3: 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.
```

8 Change Tracking

This section identifies changes that were made to the [MS-SSDPWP] protocol document between the January 2012 and February 2012 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

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 or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

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 **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

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

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

- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

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

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
2.2.1 Namespaces	400510 Removed Prefix "none" from Namespaces table.	Y	Content removed.
3.1.1 Abstract Data Model	353104 Added descriptions for Argument and State.	Y	New content added.
3.1.4.1 BeginPublish	400439 Revised the BeginPublish definition.	Y	Content removed.
3.1.4.1.2.2 BeginPublishResponse	353434 Added description for BeginPublishResponse section.	Y	Content updated.
3.1.4.2.2 Elements	353434 Updated description to include the CancelPublish operation.	Y	Content updated.
3.1.4.5 PublishData	400493 Revised description of PublishData.	Y	New content added.
3.1.4.6.2.2 PublishScriptResponse	353488 Added the description of the PublishScriptResponse element.	Y	New content added.

9 Index