[MS-FORMS]:
Forms Service Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability
06/27/2008	1.0	Major	Revised and edited the technical content
12/12/2008	1.01	Editorial	Revised and edited the technical content
07/13/2009	1.02	Major	Revised and edited the technical content
08/28/2009	1.03	Editorial	Revised and edited the technical content
11/06/2009	1.04	Editorial	Revised and edited the technical content
02/19/2010	2.0	Editorial	Revised and edited the technical content
03/31/2010	2.01	Editorial	Revised and edited the technical content
04/30/2010	2.02	Editorial	Revised and edited the technical content
06/07/2010	2.03	Editorial	Revised and edited the technical content
06/29/2010	2.04	Minor	Clarified the meaning of the technical content.
07/23/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	3.0	Major	Significantly changed the technical content.
04/11/2012	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	3.1	Minor	Clarified the meaning of the technical content.
09/12/2012	3.1	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2012	3.2	Minor	Clarified the meaning of the technical content.

Table of Contents

1 Introduction

The Forms Service Protocol enables a client to get a list of forms from a protocol server and to get individual forms from that list. A form is an XML structure with elements describing user interface elements that are displayed on a client-side Web page. The definition, display, and use of a form are outside the scope of this document.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

GUID
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)

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

absolute URL
back-end database server
content database
document
form
item
item identifier
list
list identifier
list item
page
path component
Simple Object Access Protocol (SOAP)
site
SOAP action
SOAP body
SOAP fault
store-relative form
uncustomized
Uniform Resource Locator (URL)
Web Part
Web Part zone
Web Services Description Language (WSDL)
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 Specifications documentation do not include a publishing year because links are to the latest version of the technical 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.

[MS-WSSCAML] Microsoft Corporation, "Collaborative Application Markup Language (CAML) Structure Specification".

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

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

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

[SOAP1.2/1] Gudgin, M., Hadley, M., Mendelsohn, N., 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

[SOAP1.2/2] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 2: Adjuncts", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part2-20030624

[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

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

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

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

1.2.2 Informative References

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

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

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

1.3 Overview

This protocol enables clients to obtain a list of **forms (2)** contained in a **list (1)** and to obtain information about individual forms (2). This protocol follows a straightforward request-response pattern.

1.4 Relationship to Other Protocols

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

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

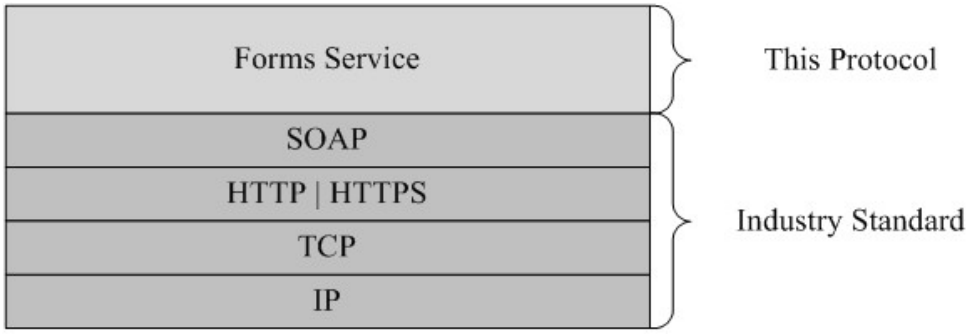

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

This protocol operates against a **site (2)** that is identified by a **URL** that is known by protocol clients. The protocol server endpoint is formed by appending "/_vti_bin/Forms.asmx" to the URL of the site, for example, http://www.contoso.com/Repository/_vti_bin/Forms.asmx.

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

1.6 Applicability Statement

This protocol enables protocol clients to obtain a list of forms (2) contained in a list (1) and to obtain information about individual forms (2).

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following area:

Supported transports: This protocol uses multiple transports with SOAP as described in section 2.1.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Protocol servers MUST support SOAP over HTTP. Protocol servers SHOULD additionally support SOAP over HTTPS for securing communication with protocol clients.

Protocol messages MUST be formatted as specified either in [SOAP1.1] section 4, SOAP Envelope, or in [SOAP1.2/1] SOAP Message Construct section 5. Protocol server faults MUST be returned either using HTTP status codes as specified in [RFC2616] Status Code Definitions section 10, or using **SOAP faults** as specified either in [SOAP1.1] SOAP Fault section 4.4, or in [SOAP1.2/1] SOAP Fault section 5.

2.2 Common Message Syntax

This section contains common definitions that are used by this protocol. The syntax of the definitions uses **XML schema**, as specified in [XMLSCHEMA1] and [XMLSCHEMA2], and **WSDL**, as specified in [WSDL].

2.2.1 Namespaces

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

Prefix	Namespace URI	Reference
soap	http://schemas.xmlsoap.org/wsdl/soap/	[SOAP1.1]
tns	http://schemas.microsoft.com/sharepoint/soap/	
s	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1] and [XMLSCHEMA2]
soap12	http://schemas.xmlsoap.org/wsdl/soap12/	[SOAP1.2/1] and [SOAP1.2/2]
wsdl	http://schemas.xmlsoap.org/wsdl/	[WSDL]

2.2.2 Messages

This specification does not define any common WSDL message definitions.

2.2.3 Elements

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

2.2.4 Complex Types

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

Complex type	Description
SOAPFaultDetails	Defines a SOAP fault.

2.2.4.1 SOAPFaultDetails

This complex type defines a SOAP fault as follows:

```
<s:complexType name="SOAPFaultDetails">
   <s:sequence>
      <s:element name="errorstring" type="s:string"/>
      <s:element name="errorcode" type="s:string" minOccurs="0"/>
   </s:sequence>
</s:complexType>
```

errorstring: A human-readable text explaining the application-level fault.

errorcode: The hexadecimal representation of a 4-byte result code. The format of the string MUST be 0xAAAAAAAA.<1>

2.2.5 Simple Types

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

2.2.6 Attributes

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

2.2.7 Groups

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

2.2.8 Attribute Groups

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

3 Protocol Details

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

Except where specified, protocol clients SHOULD interpret HTTP status codes returned by the protocol server as specified in [RFC2616] Status Code Definitions section 10.

This protocol enables protocol servers to notify protocol clients of application-level faults by using SOAP faults. This protocol enables protocol servers to provide additional details for SOAP faults by including a **detail** element as specified in either [SOAP1.1] SOAP Fault section 4.4, or [SOAP1.2/1] SOAP Fault section 5.4, that conforms to the XML schema of the **SOAPFaultDetails** complex type specified in **SOAPFaultDetails** (section 2.2.4.1). Except where specified, these SOAP faults are not significant for interoperability, and protocol clients can interpret them in an implementation-specific manner.

This protocol enables protocol servers to perform implementation-specific authorization checks and notify protocol clients of authorization faults, using either HTTP status codes or SOAP faults as specified previously in this section.

3.1 Forms Service Protocol Server Details

The following diagram describes the communication between the protocol client and the protocol server.

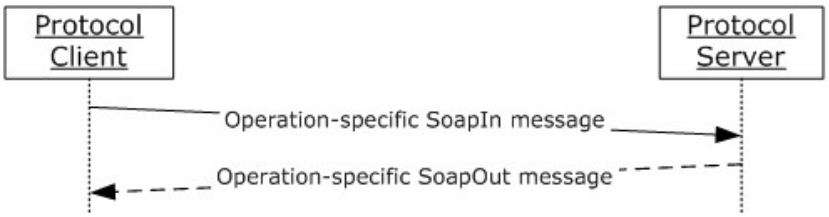

Figure 2: Sequence of methods for this protocol

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.

The protocol provides access to form (2) data stored in the **back-end database server**. The protocol does not create or modify forms (2), but enables listing and retrieval of form (2) data. The service uses the following types of data.

- **Lists:** A set of information about all lists (1) in a **content database**. Each entry has a **list identifier** and is represented by a **store-relative form** URL.
- **Items**: A set of information about all **items** in a content database. These entries have **item identifiers**.

- **Page:** An HTML **document** that may contain dynamic content such as a **Web Part** that is interpreted before being displayed in a client application.
- **Form:** A **page** that enables the creation, viewing, or editing of **list items**.

3.1.2 Timers

None.

3.1.3 Initialization

None.

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
GetForm	This operation is used to get information about a single form (2).
GetFormCollection	This operation is used to get information about all forms (2) on a particular list (1).

3.1.4.1 GetForm

This operation obtains information about a form (2). The request requires the name of the form (2) and the URL of a list (1). This operation is defined as follows:

```
<wsdl:operation name="GetForm">
    <wsdl:input message="tns:GetFormSoapIn" />
    <wsdl:output message="tns:GetFormSoapOut" />
</wsdl:operation>
```

The protocol client sends a **GetFormSoapIn** request message, and the protocol server responds with a **GetFormSoapOut** response message, as follows:

The protocol server MUST return a SOAP fault if the **listName** specified in the **GetForm** element does not exist. The detail error string SHOULD contain an error message to present to the user.

The server MUST return a SOAP fault if the **formUrl** specified in the **GetForm** element does not exist.

3.1.4.1.1 Messages

The following table summarizes the set of WSDL message definitions that are specific to this operation.

Message	Description
GetFormSoapIn	Contains the **GetForm** operation request with the parameters provided by the protocol client.
GetFormSoapOut	Contains a **GetFormResponse** element.

3.1.4.1.1.1 GetFormSoapIn

This structure contains the **GetForm** operation request with the parameters provided by the protocol client.

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

```
http://schemas.microsoft.com/sharepoint/soap/GetForm
```

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

3.1.4.1.1.2 GetFormSoapOut

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

```
http://schemas.microsoft.com/sharepoint/soap/GetForm
```

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

3.1.4.1.2 Elements

The following table summarizes the XML schema element definitions that are specific to this operation.

Element	Description
GetForm	Contains the parameters for the **GetForm** operation.
GetFormResponse	Contains the return value for the **GetForm** operation.

3.1.4.1.2.1 GetForm

The **GetForm** structure contains the parameters for the **GetForm** operation. This structure is defined as follows:

```
<s:element name="GetForm">
  <s:complexType>
    <s:sequence>
      <s:element name="listName" type="s:string"/>
      <s:element name="formUrl" type="s:string"/>
    </s:sequence>
  </s:complexType>
</s:element>
```

listName: A **GUID** encoded in a string or the title of a list (1) on the site (2).

formUrl: A URL that identifies the particular form (2) returned. This MUST be an **absolute URL** that SHOULD end with the hierarchical **path component** of the URL with no input component.

3.1.4.1.2.2 GetFormResponse

This structure contains the return value for the **GetForm** operation. This structure is defined as follows:

```
<s:element name="GetFormResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="GetFormResult" minOccurs="0">
        <s:complexType>
          <s:sequence>
            <s:element name="Form">
              <s:complexType>
                <s:attribute name="Type" use="required">
                  <s:simpleType>
                    <s:restriction base="s:string">
                      <s:enumeration value="DisplayForm" />
                      <s:enumeration value="EditForm" />
                      <s:enumeration value="NewForm" />
                      <s:enumeration value="NewFormDialog" />
                      <s:enumeration value="SolutionForm" />
                    </s:restriction>
                  </s:simpleType>
                </s:attribute>
                <s:attribute name="Name"
                    type="s:string"/>
                <s:attribute name="Url"
                    type="s:string" use="required"/>
                <s:attribute name="Default">
                  <s:simpleType name="TRUEFALSE">
                    <s:restriction base="s:string">
                      <s:pattern value="[Tt][Rr][Uu][Ee]|[Ff][Aa][Ll][Ss][Ee]"/>
                    </s:restriction>
                  </s:simpleType>
                </s:attribute>
                <s:attribute name="FormID"
                    type="s:string"/>
                <s:attribute name="Template"
                    type="s:string"/>
                <s:attribute name="WebPartZoneID"
                    type="s:string"/>
                <s:attribute name="SetupPath"
                    type="s:string"/>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
```

GetFormResult: A structure that holds the data returned from the WSDL operation.

Form: A complex type that holds the information of a form (2).

Type: A string indicating the general type of form (2) as specified in [MS-WSSCAML] section 2.3.1.5.

Name: The unique identifier of the form (2).

Url: The server relative URL of the page hosting the form (2).

Default: A Boolean value as specified in [MS-WSSCAML] section 2.1.12. Specifies whether the form (2) is the default form (2) of the list (1). It is an optional attribute that appears only if previously set in the XML schema of the form (2) definition.

FormId: A string that contains a non-negative integer. This SHOULD<2> be unique for each form (2) in the back-end database server. It is an optional attribute that appears only if previously set in the XML schema of the form (2) definition.

Template: The form (2) template name. It is an optional attribute that appears only if previously set in the XML schema of the form (2) definition.

WebPartZoneID: Identifier of the **Web Part zone** that contains the form (2). It is an optional attribute that appears only if previously set in the XML schema of the form (2) definition.

SetupPath: Source path to the **uncustomized** document of the form (2). It is an optional attribute that appears only if previously set in the XML schema of the form (2) definition.

3.1.4.1.3 Complex Types

None.

3.1.4.1.4 Simple Types

None.

3.1.4.1.5 Attributes

None.

3.1.4.1.6 Groups

None.

3.1.4.1.7 Attribute Groups

None.

3.1.4.2 GetFormCollection

This operation lists all of the forms (2) on a list (1). This operation is defined as follows:

```
<wsdl:operation name="GetFormCollection">
    <wsdl:input message="tns:GetFormCollectionSoapIn" />
    <wsdl:output message="tns:GetFormCollectionSoapOut" />
</wsdl:operation>
```

The protocol client sends a **GetFormCollectionSoapIn** request message, and the protocol server responds with a **GetFormCollectionSoapOut** response message, as follows:

If the list (1) specified by the **listName** in the **GetFormCollection** element is not found or the **GetFormCollection** element is empty, the protocol server MUST respond with a SOAP fault.

3.1.4.2.1 Messages

The following table summarizes the set of WSDL message definitions that are specific to this operation.

Message	Description
GetFormCollectionSoapIn	Contains a **GetFormCollection** element, as specified in section 3.1.4.2.2.1.
GetFormCollectionSoapOut	Contains a **GetFormsCollectionResponse** element, as specified in section 3.1.4.2.2.2.

3.1.4.2.1.1 GetFormCollectionSoapIn

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

```
http://schemas.microsoft.com/sharepoint/soap/GetFormCollection
```

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

3.1.4.2.1.2 GetFormCollectionSoapOut

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

```
http://schemas.microsoft.com/sharepoint/soap/GetFormCollection
```

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

3.1.4.2.2 Elements

The following table summarizes the XML schema element definitions that are specific to this operation.

Element	Description
GetFormCollection	Used to pass the parameter for the **GetFormCollection** operation.
GetFormCollectionResponse	Holds a list of **Form** elements, each of which describes a single form (2).

3.1.4.2.2.1 GetFormCollection

This structure is used to pass the parameter for the **GetFormCollection** WSDL operation. This operation is defined as follows:

```
<s:element name="GetFormCollection">
  <s:complexType>
    <s:sequence>
      <s:element name="listName" type="s:string"/>
    </s:sequence>
  </s:complexType>
</s:element>
```

listName: A GUID encoded in a string or the title of a list (1) from which to fetch forms (2).

3.1.4.2.2.2 GetFormCollectionResponse

This structure holds a list of **Form** elements, each of which describes a single form (2). This structure is defined as follows:

```
<s:element name="GetFormCollectionResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="GetFormCollectionResult">
        <s:complexType>
          <s:sequence>
            <s:element name="Forms">
              <s:complexType>
                <s:sequence>
                  <s:element name="Form" minOccurs="0"
                             maxOccurs="unbounded">
                    <s:complexType>
                      <s:attribute name="Type" use="required">
                        <s:simpleType>
                          <s:restriction base="s:string">
                            <s:enumeration value="DisplayForm" />
                            <s:enumeration value="EditForm" />
                            <s:enumeration value="NewForm" />
                            <s:enumeration value="NewFormDialog" />
                            <s:enumeration value="SolutionForm" />
                            <s:enumeration value="" />
                          </s:restriction>
                        </s:simpleType>
                      </s:attribute>
                      <s:attribute name="Url" type="s:string"
                              use="required"/>
                    </s:complexType>
                  </s:element>
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
```

GetFormCollectionResult: This structure holds the data returned from the WSDL operation..

Forms: A list of complex types **Form**. It holds the information of a list of form (2).

Form: A complex type that holds the information of a form (2).

Type: A string with the form (2) type. It MUST be the same type as the one defined in **GetFormResponse** (section 3.1.4.1.2.2).

Url: The server relative URL<3> of the page hosting the form (2).

3.1.4.2.3 Complex Types

None.

3.1.4.2.4 Simple Types

None.

3.1.4.2.5 Attributes

None.

3.1.4.2.6 Groups

None.

3.1.4.2.7 Attribute Groups

None.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

4 Protocol Examples

This section provides some example client-server exchanges. White space has been added to improve readability.

4.1 GetFormCollection

A client-server exchange for the **GetFormCollection** operation resembles the following example.

Client to server:

```
<?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>
    <GetFormCollection
         xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <listName>Custom Test</listName>
    </GetFormCollection>
  </soap:Body>
</soap:Envelope>
```

Server response:

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope
       xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <soap:Body>
        <GetFormCollectionResponse
            xmlns="http://schemas.microsoft.com/sharepoint/soap/">
            <GetFormCollectionResult>
                <Forms>
                   <Form Url="Lists/Custom Test/DispForm.aspx"
                         Type="DisplayForm" />
                   <Form Url="Lists/Custom Test/EditForm.aspx"
                         Type="EditForm" />
                   <Form Url="Lists/Custom Test/NewForm.aspx"
                         Type="NewForm" />
                </Forms>
             </GetFormCollectionResult>
       </GetFormCollectionResponse>
      </soap:Body>
  </soap:Envelope>
```

4.2 GetForm

A client-server exchange for the **GetForm** operation resembles the following example.

Client request:

```
<?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>
    <GetForm xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <listName>{FDA9C196-8655-45C4-91BB-2B42682CD278}</listName>
      <formUrl>http://office/Lists/Custom Test/NewForm.aspx</formUrl>
    </GetForm>
  </soap:Body>
</soap:Envelope>
```

Server response:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
       <GetFormResponse xmlns="http://schemas.microsoft.com/sharepoint/soap/">
             <GetFormResult>
                 <Form Type="NewForm"
                       Url="NewForm.aspx"
                       SetupPath="pages\form.aspx"
                       WebPartZoneID="Main" />
             </GetFormResult>
        </GetFormResponse>
   </soap:Body>
</soap:Envelope>
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Full WSDL

For ease of implementation, the full WSDL and schema are provided in this appendix.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://schemas.microsoft.com/sharepoint/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
targetNamespace="http://schemas.microsoft.com/sharepoint/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://schemas.microsoft.com/sharepoint/soap/">
      <s:import namespace="http://www.w3.org/2001/XMLSchema" />
<s:element name="GetForm">
  <s:complexType>
    <s:sequence>
      <s:element name="listName" type="s:string"/>
      <s:element name="formUrl" type="s:string"/>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetFormResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="GetFormResult" minOccurs="0">
        <s:complexType>
          <s:sequence>
            <s:element name="Form">
              <s:complexType>
                <s:attribute name="Type" use="required">
                  <s:simpleType>
                    <s:restriction base="s:string">
                      <s:enumeration value="DisplayForm" />
                      <s:enumeration value="EditForm" />
                      <s:enumeration value="NewForm" />
                      <s:enumeration value="NewFormDialog" />
                      <s:enumeration value="SolutionForm" />
                    </s:restriction>
                  </s:simpleType>
                </s:attribute>
                <s:attribute name="Name"
                    type="s:string"/>
                <s:attribute name="Url"
                    type="s:string" use="required"/>
                <s:attribute name="Default">
                  <s:simpleType name="TRUEFALSE">
                    <s:restriction base="s:string">
                      <s:pattern
                          value="[Tt][Rr][Uu][Ee]|[Ff][Aa][Ll][Ss][Ee]"/>
                    </s:restriction>
                  </s:simpleType>
                </s:attribute>
                <s:attribute name="FormID"
                    type="s:string"/>
                <s:attribute name="Template"
                    type="s:string"/>
                <s:attribute name="WebPartZoneID"
```

```
                    type="s:string"/>
                <s:attribute name="SetupPath"
                    type="s:string"/>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetFormCollection">
  <s:complexType>
    <s:sequence>
      <s:element name="listName" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetFormCollectionResponse">
  <s:complexType>
    <s:sequence>
      <s:element name="GetFormCollectionResult">
        <s:complexType>
          <s:sequence>
            <s:element name="Forms">
              <s:complexType>
                <s:sequence>
                  <s:element name="Form" minOccurs="0"
                             maxOccurs="unbounded">
                    <s:complexType>
                      <s:attribute name="Type" use="required">
                        <s:simpleType>
                          <s:restriction base="s:string">
                            <s:enumeration value="DisplayForm" />
                            <s:enumeration value="EditForm" />
                            <s:enumeration value="NewForm" />
                            <s:enumeration value="NewFormDialog" />
                            <s:enumeration value="SolutionForm" />
                            <s:enumeration value="" />
                          </s:restriction>
                        </s:simpleType>
                      </s:attribute>
                      <s:attribute name="Url" type="s:string"
                              use="required"/>
                    </s:complexType>
                  </s:element>
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="detail">
  <s:complexType>
    <s:sequence>
      <s:element name="errorString" type="s:string"/>
```

```
      <s:element name="errorCode" type="s:string" minOccurs="0"/>
    </s:sequence>
  </s:complexType>
</s:element>
   </s:schema>
   <s:schema xmlns:s="http://www.w3.org/2001/XMLSchema" targetNamespace="
http://schemas.microsoft.com/sharepoint/soap">
     <s:complexType name="SOAPFaultDetails">
       <s:sequence>
         <s:element name="errorstring" type="s:string"/>
         <s:element name="errorcode" type="s:string" minOccurs="0"/>
        </s:sequence>
     </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="GetFormCollectionSoapIn">
    <wsdl:part name="parameters" element="tns:GetFormCollection" />
  </wsdl:message>
  <wsdl:message name="GetFormCollectionSoapOut">
    <wsdl:part name="parameters" element="tns:GetFormCollectionResponse" />
  </wsdl:message>
  <wsdl:message name="GetFormSoapIn">
    <wsdl:part name="parameters" element="tns:GetForm" />
  </wsdl:message>
  <wsdl:message name="GetFormSoapOut">
    <wsdl:part name="parameters" element="tns:GetFormResponse" />
  </wsdl:message>
  <wsdl:portType name="FormsSoap">
    <wsdl:operation name="GetFormCollection">
      <wsdl:input message="tns:GetFormCollectionSoapIn" />
      <wsdl:output message="tns:GetFormCollectionSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="GetForm">
      <wsdl:input message="tns:GetFormSoapIn" />
      <wsdl:output message="tns:GetFormSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="FormsSoap" type="tns:FormsSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetFormCollection">
      <soap:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/GetFormCollection" style="document"
/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetForm">
      <soap:operation soapAction="http://schemas.microsoft.com/sharepoint/soap/GetForm"
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="FormsSoap12" type="tns:FormsSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetFormCollection">
      <soap12:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/GetFormCollection" style="document"
/>
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetForm">
      <soap12:operation soapAction="http://schemas.microsoft.com/sharepoint/soap/GetForm"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```

7 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® Office 2003
- The 2007 Microsoft® Office system
- Microsoft® Office 2010 suites
- Microsoft® Office 2013
- Windows® SharePoint® Services 2.0
- Windows® SharePoint® Services 3.0
- Microsoft® SharePoint® Foundation 2010
- Microsoft® SharePoint® Foundation 2013

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 2.2.4.1: Windows SharePoint Services 3.0 does not return a result code.

<2> Section 3.1.4.1.2.2: Windows SharePoint Services 3.0 returns the **FormId** of the default form that is the same type as the form specified by using Form Url in the SOAP request.

<3> Section 3.1.4.2.2.2: Windows SharePoint Services 3.0 returns the URLs relative to the list (1) from which the forms (2) are fetched.

8 Change Tracking

This section identifies changes that were made to the [MS-FORMS] protocol document between the September 2012 and October 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
3.1.4.1 GetForm	Updated schema to include namespace prefixes.	N	Content updated.
3.1.4.2 GetFormCollection	Updated schema to include namespace prefixes.	N	Content updated.

9 Index

T

V

W