[MS-FORMS]:

Forms Service Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **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 may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks**. The names of companies and products contained in this documentation may 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, e-mail addresses, logos, people, places, and events 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 specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do 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 are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release version of this technology. This Open Specification is final documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional

Preliminary

development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Preliminary

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1		Initial Availability
6/27/2008	1.0	Major	Revised and edited the technical content
12/12/2008	1.01	Editorial	Revised and edited the technical content
7/13/2009	1.02	Major	Revised and edited the technical content
8/28/2009	1.03	Editorial	Revised and edited the technical content
11/6/2009	1.04	Editorial	Revised and edited the technical content
2/19/2010	2.0	Editorial	Revised and edited the technical content
3/31/2010	2.01	Editorial	Revised and edited the technical content
4/30/2010	2.02	Editorial	Revised and edited the technical content
6/7/2010	2.03	Editorial	Revised and edited the technical content
6/29/2010	2.04	Minor	Clarified the meaning of the technical content.
7/23/2010	2.04	No Change	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	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.
3/18/2011	2.04	No Change	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	2.04	No Change	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	3.0	Major	Significantly changed the technical content.
4/11/2012	3.0	No Change	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	3.1	Minor	Clarified the meaning of the technical content.
9/12/2012	3.1	No Change	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	3.2	Minor	Clarified the meaning of the technical content.
2/11/2013	3.2	No Change	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	3.2	No Change	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	3.3	Minor	Clarified the meaning of the technical content.
2/10/2014	3.3	No Change	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	3.3	No Change	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	3.3	No Change	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	3.4	Minor	Clarified the meaning of the technical content.
3/16/2015	4.0	Major	Significantly changed the technical content.

Preliminary

Table of Contents

Preliminary

Preliminary

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 [RFC2119]. Sections 1.5 and 1.9 are also normative but do not contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are specific to this document:

absolute URL: The full Internet address of a page or other World Wide Web resource. The absolute URL includes a protocol, such as "http," a network location, and an optional path and file name — for example, http://www.treyresearch.net/.

back-end database server: A server that hosts data, configuration settings, and stored procedures that are associated with one or more applications.

content database: A database that is stored on a **back-end database server** and contains stored procedures, site collections, and the contents of those site collections.

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

form: A document with a set of controls into which users can enter information. Controls on a form can be bound to elements in the data source of the form, such as fields and groups. See also bind.

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

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

Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS): An extension of **HTTP** that securely encrypts and decrypts webpage requests.

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

item identifier: An integer that uniquely identifies an item in a SharePoint list.

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

list identifier: A GUID that is used to identify a **list** in a site collection.

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

Preliminary

page: A file that consists of HTML and can include references to graphics, scripts, or dynamic content such as Web Parts.

path component: Data that identifies a resource within the scope of a scheme and authority in a URI, as described in [RFC3986].

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

SOAP: A lightweight protocol for exchanging structured information in a decentralized, distributed environment. **SOAP** uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation-specific semantics. SOAP 1.2 supersedes SOAP 1.1. See [SOAP1.2-1/2003].

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

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

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

store-relative form: See store-relative URL.

uncustomized: A condition of a document whose content is stored in a location other than the content database. If a document is uncustomized, the front-end web server determines the location of the content by using the SetupPath value for the document. Also referred to as ghosted.

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

Web Part: A reusable component that contains or generates web-based content such as XML, HTML, and scripting code. It has a standard property schema and displays that content in a cohesive unit on a webpage. See also Web Parts Page.

Web Part zone: A structured HTML section of a Web Parts Page that contains zero or more Web Parts and can be configured to control the organization and format of those Web Parts.

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.

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.

Preliminary

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

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.

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

[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.rfc-editor.org/rfc/rfc2616.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.rfc-editor.org/rfc/rfc2818.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., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

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

1.2.2 Informative References

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, https://www2.opengroup.org/ogsys/catalog/c706

[RFC1738] Berners-Lee, T., Masinter, L., and McCahill, M., Eds., "Uniform Resource Locators (URL)", RFC 1738, December 1994, http://www.ietf.org/rfc/rfc1738.txt

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, http://www.ietf.org/rfc/rfc3986.txt

Preliminary

[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005, http://www.ietf.org/rfc/rfc4122.txt

[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

[SOAP1.2-1/2007] Gudgin, M., Hadley, M., Mendelsohn, N., et al., "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)", W3C Recommendation 27, April 2007, http://www.w3.org/TR/2007/REC-soap12-part1-20070427/

[XMLNS-2ED] World Wide Web Consortium, "Namespaces in XML 1.0 (Second Edition)", August 2006, http://www.w3.org/TR/2006/REC-xml-names-20060816/

1.3 Overview

This protocol enables clients to obtain a list of **forms** contained in a **list** and to obtain information about individual forms. 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:

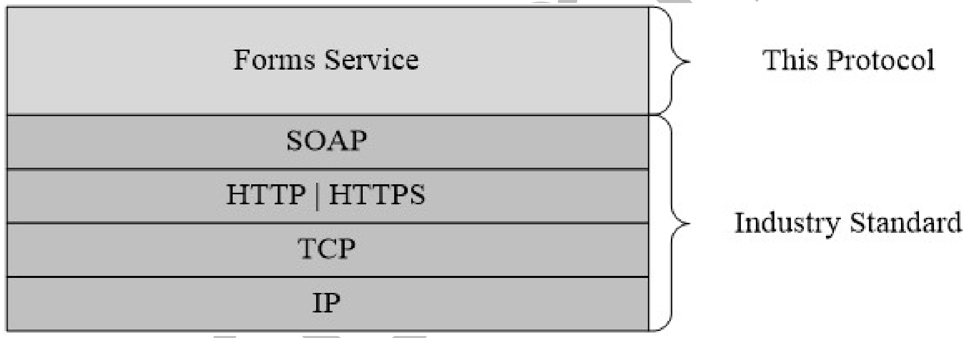

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

This protocol operates against a **site** 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 contained in a list and to obtain information about individual forms.

Preliminary

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.

Preliminary

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, as described in [RFC2818].

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.

Preliminary

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.

Preliminary

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.

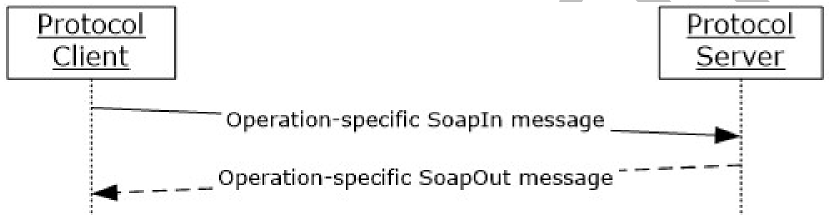
Protocol Client
Protocol Server
Operation-specific SoapIn message
Operation-specific SoapOut message

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

- **Lists:** A set of information about all lists 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**.

Preliminary

- **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.
GetFormCollection	This operation is used to get information about all forms on a particular list.

3.1.4.1 GetForm

This operation obtains information about a form. The request requires the name of the form and the URL of a list. 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 (section 3.1.4.1.2.1) 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, as specified in section 3.1.4.1.2.2.

Preliminary

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, as specified in section 3.1.4.1.2.1.

3.1.4.1.1.2 GetFormSoapOut

The SOAP body contains a **GetFormResponse** element, as specified in section 3.1.4.1.2.2.

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 on the site.

formUrl: A URL that identifies the particular form 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>
```

Preliminary

```
      <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>
                    <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>
```

Preliminary

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

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

Type: A string indicating the general type of form 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. 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 on a list. 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 specified by the **listName** in the **GetFormCollection** element (section 3.1.4.2.2.1) is not found or the **GetFormCollection** element is empty, the protocol server MUST respond with a SOAP fault.

3.1.4.2.1 Messages

Preliminary

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 **GetFormCollectionResponse** 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, as specified in section 3.1.4.2.2.1.

3.1.4.2.1.2 GetFormCollectionSoapOut

The SOAP body contains a **GetFormCollectionResponse** element, as specified in section 3.1.4.2.2.2.

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.

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 from which to fetch forms.

3.1.4.2.2.2 GetFormCollectionResponse

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

Preliminary

```
<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.

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

Type: A string with the form 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.

3.1.4.2.3 Complex Types

None.

Preliminary

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.

Preliminary

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.

Preliminary

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>
```

Preliminary

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

Preliminary

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>
                    <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: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
- Windows 8.1 Update
- Microsoft Office 2016 Preview

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 from which the forms are fetched.

Preliminary

8 Change Tracking

This section identifies changes that were made to this document since the last release. 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.
- The removal of a document from the documentation set.

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 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 changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

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.
- Obsolete document removed.

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

Preliminary

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 dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
7 Appendix B: Product Behavior	Updated list of supported products.	Y	Content updated due to protocol revision.

Preliminary

9 Index

Preliminary

Preliminary