[MS-FSPP]:
Forms Services Proxy Web Service Protocol Specification

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's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). 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.
- **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.

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
01/16/2009	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	Minor	Updated 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	Editorial	Changed language and formatting in the technical content.
07/23/2010	2.05	Minor	Clarified the meaning of the technical content.
09/27/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	2.6	Minor	Clarified the meaning of the technical content.

Table of Contents

1 Introduction

This document specifies the Forms Services Proxy Web Service Protocol, which allows the protocol client to call a service operation through a single centralized service located on a known protocol server. The protocol server acts as a bridge between the protocol client and a target Web service by forwarding an authenticated message from the protocol client to a target Web service operation.

1.1 Glossary

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

authentication
authorization
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
Unicode
XML

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

browser-enabled form template
data adapter
form template (.xsn) file
Internationalized Resource Identifier (IRI)
path segment
Simple Object Access Protocol (SOAP)
site
site collection
SOAP action
SOAP body
SOAP fault
SOAP header
Status-Code
Uniform Resource Locator (URL)
Universal Data Connection (.udc, .udcx) file
Web service
Web Services Description Language (WSDL)
XML namespace
XML schema definition (XSD)

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

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-IPFF] Microsoft Corporation, "InfoPath Form Template Format"

[MS-IPFF2] Microsoft Corporation, "InfoPath Form Template Format Version 2"

[MS-UDCX] Microsoft Corporation, "Universal Data Connection 2.0 XML File Format"

[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

[RFC3987] Duerst, M., and Suignard, M., "Internationalized Resource Identifiers (IRIs)," RFC 3987, January 2005, http://www.ietf.org/rfc/rfc3987.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

[WSSE 1.0] Nadalin, A., Kaler, C., Hallam-Baker, P., and Monzillo, R., Eds., "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", OASIS Standard 200401, March 2004, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[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 Protocol Overview (Synopsis)

The Forms Services Proxy Web Service Protocol specifies how a protocol client that is currently processing a **form template (.xsn) file** can request the protocol server to forward a **SOAP body** to a target **Web service** as described either in [SOAP1.1] or [SOAP1.2/1]. The protocol server creates a **Simple Object Access Protocol (SOAP)** message using the SOAP body received from

the protocol client, and forwards this message to the target Web service using **Hypertext Transfer Protocol (HTTP)** or **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)**. This protocol enables a protocol server to provide the credentials used to access the target Web service. This protocol contains two messages: the request message, specified in section 3.1.4.1.1.1, and the response message, specified in section 3.1.4.1.1.2.

This document specifies the messages between the protocol client and the proxy on the protocol server as well as the **SOAP header** of the SOAP message to forward to the target Web service. The following figure illustrates the protocol.

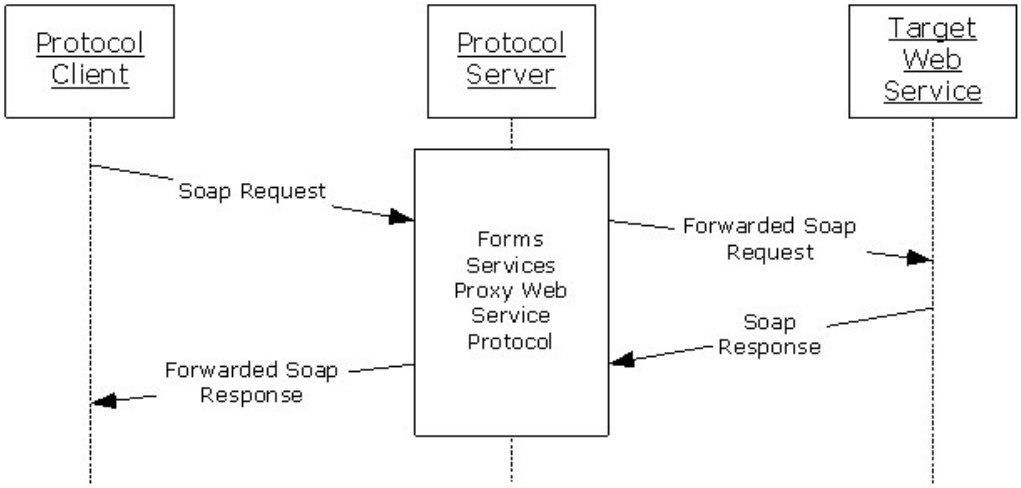

Figure 1: Protocol workflow

This protocol does not specify any behavior of the target Web service beyond the preconditions in section 1.5.

1.4 Relationship to Other Protocols

This protocol uses the SOAP message protocol for formatting request and response messages, as described in [SOAP1.1] or as described in [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:

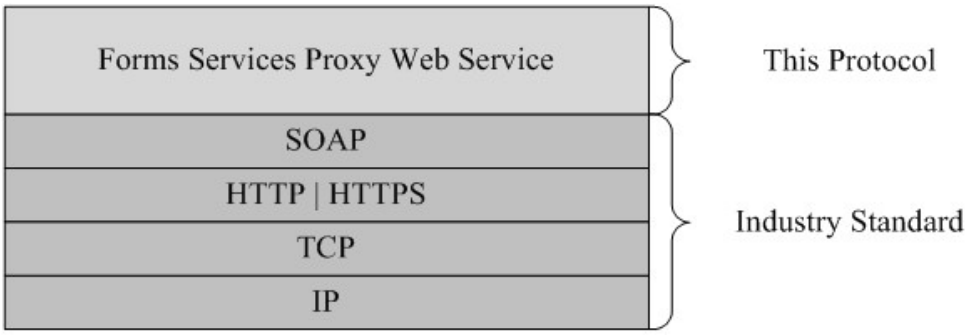

Figure 2: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

The Forms Services Proxy Web Service Protocol operates against a **site (1)** that is identified by a **Uniform Resource Locator (URL)** that is known by protocol clients. The protocol server endpoint is formed by appending "_vti_bin/FormsServiceProxy.asmx" to the URL of the site (1), for example http://www.contoso.com/Repository/_vti_bin/FormsServiceProxy.asmx.

This protocol assumes that **authentication (1)** has been performed by the underlying protocols.

There are also preconditions specific to the Forms Services Proxy Web Service Protocol that need to be met before this protocol can be used successfully.

This protocol assumes that both the protocol client and protocol server have copies of a form template (.xsn) file resource, which is addressable via a URL. This protocol does not specify how the protocol client and protocol server obtain their respective copies of this resource.

The protocol client is also expected to be processing a **Universal Data Connection (.udc, .udcx) file** (UDC file) that is referenced by a **data adapter** in a form template (.xsn) file. This protocol assumes that both the protocol client and protocol server have copies of this UDC file. This protocol assumes that this UDC file has a **Type** attribute on the **Type** element equal to "WebService" as described in [MS-UDCX], section 2.3.4. This protocol does not specify how the protocol client and protocol server obtain their respective copies of this UDC file.

This protocol assumes that the target Web service operation identified in this UDC file describes the **style** attribute for the **soap:operation** element as "document" ([WSDL], section 3.4), and the **use** attribute for the **soap:body** element as "literal" ([WSDL], section 3.5). This protocol assumes the protocol client can construct a valid request SOAP body for the target Web service operation.

1.6 Applicability Statement

The Forms Services Proxy Web Service Protocol is applicable when the following conditions are met:

- The protocol client needs to perform a query or submit to a Web service referenced by a data adapter which is specified by a UDC file.
- The UDC file contains a **UseFormsServiceProxy** attribute with a value of "true" as specified in section 3.1.4.1.1.1.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports:** This document covers versioning issues with SOAP as specified in section 2.1.

Protocol Versions: This protocol refers to different file format specifications, [MS-IPFF] and [MS-IPFF2], both of which define the structure of a valid form template (.xsn) file. In cases where both specifications are cited as references, the **SolutionFormatVersion** attribute of the **xDocumentClass** element, as described in [MS-IPFF2] section 2.2.1.2.1, specifies whether to use the InfoPath Form Template Format, as described in [MS-IPFF], or the InfoPath Form Template Format Version 2, as described in [MS-IPFF2].

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 clients.

Protocol messages MUST be formatted as specified in either [SOAP1.1], section 4, or in [SOAP1.2/1], section 5. Protocol server faults MUST be returned either using HTTP **Status-Codes** as specified in [RFC2616], section 10, or using **SOAP faults** as specified either in [SOAP1.1], section 4.4, or in [SOAP1.2/1], section 5.4.

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 **WSDL** as defined in [WSDL].

Messages sent from the protocol client to the protocol server are specified in section 3.1.4.1.2.1. Messages sent from the protocol server to the protocol client are specified in section 3.1.4.1.2.2.

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/office/infopath/2007/formsServicesProxy	
s	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1] [XMLSCHEMA2]
soap12	http://schemas.xmlsoap.org/wsdl/soap12/	[SOAP1.2/1] [SOAP1.2/2]
(none)	http://schemas.microsoft.com/office/infopath/2007/formsServicesProxy	
wsdl	http://schemas.xmlsoap.org/wsdl/	[WSDL]
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	[WSSE 1.0]

2.2.2 Messages

None.

2.2.3 Elements

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

2.2.4 Complex Types

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

2.2.5 Simple Types

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

2.2.6 Attributes

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

2.2.7 Groups

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

2.2.8 Attribute Groups

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

3 Protocol Details

The client side of this protocol is simply 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], section 10.

This protocol allows protocol servers to notify protocol clients of application-level faults using SOAP faults. Except where specified, these SOAP faults are not significant for interoperability, and protocol clients can interpret them in an implementation-specific manner.

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

The server side of this protocol is specified as follows.

3.1 Server Details

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 server processes request messages by identifying a target Web service and target Web service operation and then forwarding a SOAP body received from the protocol client to the target Web service.

The following figure illustrates the relationship between objects the protocol server uses to process messages in this protocol. Section 3.1.4.1.1.1 specifies how the protocol server finds and validates these objects and constructs the SOAP message to send to the target Web service.

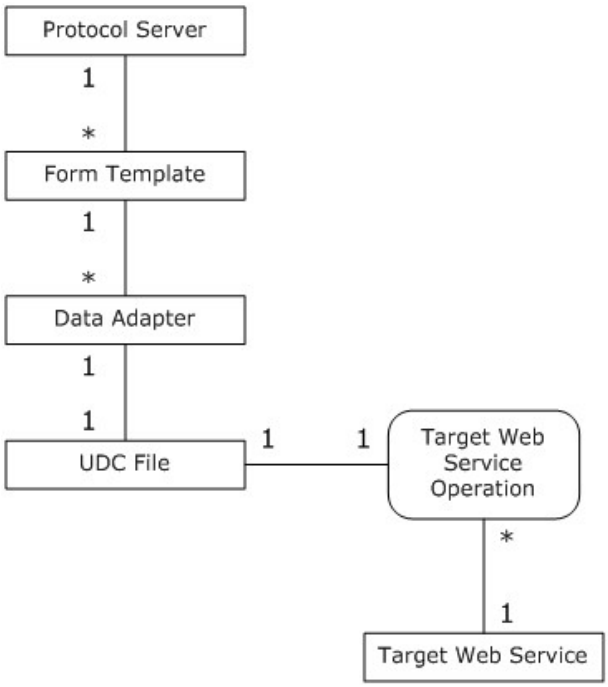

Figure 3: Abstract data model

Form Template: The protocol server maintains a mapping using both a URL and a version string to identify at most one form template (.xsn) file. The protocol uses this mapping and the **url** and **version** elements of the request message specified in section 3.1.4.1.2.1 to identify the form template (.xsn) file used for the request. Any authorization for this form template (.xsn) file is implementation-specific.

Data Adapter: The protocol server uses the **adapterName** element of the request message specified in section 3.1.4.1.2.1 to find a matching data adapter in the form template (.xsn) file.

UDC File: The protocol server uses properties of this data adapter to find a UDC file using one of the following mappings:

- A mapping using a URL to identify a UDC file.
- A mapping using a file name to identify a UDC file in an implementation-specific store.

Target Web Service: The protocol server uses information in the UDC file to identify the target Web service and target Web service operation.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Message Processing Events and Sequencing Rules

The following table summarizes the operation of this protocol.

Operation	Description
ForwardSoapRequest	Forward a Web service request from the protocol client to a target Web service operation via the protocol server.

3.1.4.1 ForwardSoapRequest

This operation is used to forward a Web service request from the protocol client to a target Web service operation via the protocol server.

```
<wsdl:operation name="ForwardSoapRequest">
    <wsdl:input message="ForwardSoapRequestSoapIn" />
    <wsdl:output message="ForwardSoapRequestSoapOut" />
</wsdl:operation>
```

A protocol client sends a **ForwardSoapRequestSoapIn** request message to a protocol server to initiate the protocol. The protocol server URL is discovered as described in section 1.5.

The protocol server MUST respond to a **ForwardSoapRequestSoapIn** message from a protocol client as follows:

1. The protocol server MUST find a form template (.xsn) file using the following two elements in the request message:

   - The **url** element which MUST identify the form template (.xsn) file.
   - The **version** element which MUST equal the value of the **solutionVersion** attribute as specified in [MS-IPFF] section 2.2.20 and [MS-IPFF2] section 2.2.1.2.1 of the form template (.xsn) file as specified in section 3.1.4.1.2.1.

     The **url** and **version** elements are uniquely mapped to a form template (.xsn) file as specified in section 3.1.1<1>.

2. The protocol server MUST find a **connectoid** element as specified in [MS-IPFF] section 2.2.147.30 and [MS-IPFF2] section 2.2.2.2.23 in this form template (.xsn) file that is identified by the **adapterName** element in the request message as follows:

   - The protocol server MUST find a **webServiceAdapter** element in the previously identified form template (.xsn) file where the **name** attribute as specified in [MS-IPFF] section 2.2.39 and [MS-IPFF2] section 2.2.1.2.20, is equal to the value of the **adapterName** element in the request message.
   - The protocol server MUST find a **webServiceAdapterExtension** element in this form template (.xsn) file where the **ref** attribute as specified in [MS-IPFF] section 2.2.147.33 and [MS-IPFF2] section 2.2.2.2.26, equals the value of the **adapterName** element in the request message.
   - This **webServiceAdapterExtension** element MUST have a child **connectoid** element as specified in [MS-IPFF] section 2.2.147.30 and [MS-IPFF2] section 2.2.2.2.23.

3. The protocol server MUST find a UDC file from this **connectoid** element as follows:

- If the **connectionLinkType** attribute on this **connectoid** element has the value "relative", then the protocol server MUST find a UDC file using the mapping from URL to UDC file specified in section 3.1.1. The protocol server MUST create the URL to the UDC file by appending the value of the **connectoid** element's **source** attribute to the URL of the **site collection** where the form template (.xsn) file is located. The protocol server MUST verify this URL identifies a valid UDC file as specified in [MS-UDCX].

- Otherwise, the protocol server MUST verify the **connectionLinkType** attribute on this **connectoid** element has the value "store". Then the protocol server MUST extract the rightmost **path segment** of this **connectoid** element's **source** attribute, and then use this value to identify a UDC file using the file name to UDC file mapping specified in section 3.1.1.

4. After retrieving a UDC file, the protocol server MUST identify the target Web service and target Web service operation as follows:

   - The server MUST identify a command (either a **SelectCommand** element as specified in [MS-UDCX], section 2.3.7, or an **UpdateCommand** element as specified in [MS-UDCX], section 2.3.12) in the UDC file that identifies the target Web service. If the **webServiceAdapter** element previously found has a **submitAllowed** attribute with the value "yes" as specified in [MS-IPFF] section 2.2.39 and [MS-IPFF2] section 2.2.1.2.20, then the UDC file MUST contain an **UpdateCommand** element as specified in [MS-UDCX], section 2.3.12. Otherwise, the UDC file MUST contain a **SelectCommand** element as specified in [MS-UDCX], section 2.3.7. The protocol server MUST use this command for the following checks.

   - This command MUST have a child **ServiceUrl** element, as specified in [MS-UDCX] section 2.3.18. The value of this element MUST be a valid URL that the protocol server will use as the URL of the target Web service<2>.

   - The protocol server MUST validate that the **UseFormsServiceProxy** attribute as specified in [MS-UDCX], section 2.3.18, is present on this **ServiceUrl** element, and that the value of the attribute is "true", using a case-insensitive comparison.

   - The command MUST have a child **SoapAction** element, as specified in [MS-UDCX] section 2.3.15. The value of this element MUST be a valid SOAP action and specifies the target Web service operation.

5. The protocol server MUST create a SOAP message in the format specified in either [SOAP1.1] or [SOAP1.2/1] to send to the target Web service as follows:

   - The SOAP body of the new message MUST be set to the value of the **xmlContent** element of the request message as specified in section 3.1.4.1.2.1.

   - The SOAP action of the new message MUST be set to the target Web service operation that was identified from the UDC file in the preceding step.

   - The protocol server SHOULD include a **Security XML** element as specified in [WSSE 1.0] within the SOAP header of this SOAP message. If included, the content of this element MUST be valid according to the following **XML schema definition (XSD)**.

     The value of the **Username** element SHOULD be the unique string used by any implementation-specific authentication (1) to identify the user of the protocol client, but MAY<3> be an empty string.

```
<s:schema targetNamespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:s="http://www.w3.org/2001/XMLSchema">
```

```
    <s:element name="Security">
      <s:complexType>
        <s:all>
          <s:element ref="wsse:UsernameToken"/>
        </s:all>
      </s:complexType>
    </s:element>
    <s:element name="UsernameToken">
      <s:complexType>
        <s:all>
          <s:element ref="wsse:Username"/>
        </s:all>
      </s:complexType>
    </s:element>
    <s:element name="Username" type="s:string"/>
  </s:schema>
```

- If the UDC file contains an **Authentication** element as specified in [MS-UDCX], section 2.3.19, then the protocol server MUST use the authentication (2) method and credentials specified in that element to make the request.

6. The protocol server MUST send this SOAP message to the URL of the target Web service that was identified from the UDC file in step 4.
7. When a response is received from the target Web service, the protocol server MUST return a response message to the protocol client as specified in section 3.1.4.1.1.2.
8. If any of the preceding validation steps fail, then the protocol server MUST return a Status-Code of 4*xx* or 5*xx* as specified in [RFC2616], and MUST return a SOAP fault. The values of all elements and attributes in the SOAP fault are implementation-dependent and not specified by this protocol.

The protocol server MUST construct the **ForwardSoapRequestSoapOut** response message as follows:

- If the protocol server successfully processes the request as specified in the preceding algorithm for **ForwardSoapRequestSoapIn**, and the target Web service returns a Status-Code of 200, then the protocol server MUST return a Status-Code of 200 as specified in [RFC2616]. The response SOAP body MUST be a valid **ForwardSoapRequestResponse** element as specified in section 3.1.4.1.2.2. The **ForwardSoapRequestResult** element in the response MUST be the SOAP body returned by the target Web service operation.
- Otherwise, the protocol server MUST return a Status-Code of 4*xx* or 5*xx* as specified in [RFC2616], and MUST return a SOAP fault.
  - If a SOAP fault was returned to the protocol server by the target Web service, then the protocol server MUST return a Status-Code of 500 to the protocol client, and the value of the **faultcode** element in the SOAP fault returned to the protocol client MUST be the value of the **faultcode** element in the SOAP fault returned by the target Web service.
  - For all other failure reasons, any implementation-specific values can be returned in the SOAP fault.

3.1.4.1.1 Messages

The following WSDL message definitions are specific to this operation.

3.1.4.1.1.1 ForwardSoapRequestSoapIn

ForwardSoapRequestSoapIn is the request that a protocol client passes to the **ForwardSoapRequest** operation, as specified in section 3.1.4.1.

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

```
http://schemas.microsoft.com/office/infopath/2007/formsServicesProxy/ForwardSoapRequest
```

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

3.1.4.1.1.2 ForwardSoapRequestSoapOut

ForwardSoapRequestSoapOut contains the results obtained by the protocol server by calling the target Web service operation.

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

```
http://schemas.microsoft.com/office/infopath/2007/formsServicesProxy/ForwardSoapRequest
```

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

3.1.4.1.2 Elements

3.1.4.1.2.1 ForwardSoapRequest

ForwardSoapRequest specifies the content of a message from the protocol client to the protocol server.

```
<s:element name="ForwardSoapRequest">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="xmlContent">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element minOccurs="1" maxOccurs="1" name="version"
         type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="url"
         type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="adapterName"
         type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

xmlContent: A SOAP body for a request to be executed on the target Web service operation.

version: The version of the form template (.xsn) file the protocol client is using. This value MUST be valid according to the **xdSolutionVersion** type specified in [MS-IPFF] section 2.2.10 and [MS-

IPFF2] section 2.2.1.1.10, and MUST be equal to the **solutionVersion** attribute specified in [MS-IPFF] section 2.2.20 and [MS-IPFF2] section 2.2.1.2.1 in the form template (.xsn) file identified by the **url** element.

url: An **Internationalized Resource Identifier (IRI)** of the form template (.xsn) file that the protocol client is processing. Together the **url** and **version** elements MUST uniquely identify a form template (.xsn) file that is known by both the protocol client and protocol server as described in section 1.5. The value of this element MUST be an IRI that can be converted to an HTTP or HTTPS URL as specified in [RFC3987].

adapterName: Identifies a **webServiceAdapter** element as specified in [MS-IPFF] section 2.2.39 and [MS-IPFF2] section 2.2.1.2.20, in the form template (.xsn) file. This value MUST be a non-empty **Unicode** string that equals the value of the **name** attribute on a **webServiceAdapter** element in the form template (.xsn) file. Section 3.1.4.1.1.1 specifies how this element is used in identifying the target Web service and target Web service operation.

3.1.4.1.2.2 ForwardSoapRequestResponse

ForwardSoapRequestResponse specifies the content of a message from the protocol server to the protocol client.

```
<s:element name="ForwardSoapRequestResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="ForwardSoapRequestResult">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
```

ForwardSoapRequestResult: The SOAP body returned from the target Web service operation after the target Web service has executed the forwarded request.

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.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

4 Protocol Examples

This section contains examples of request and response messages sent by the protocol client, protocol server, and the target Web service using the Forms Services Proxy Web Service Protocol. In this sample scenario, the protocol client needs to query the **HelloWorld** operation on a Web service located at http://www.contoso.com/Service.asmx. The example messages use the XML namespace prefixes specified in section 2.2.1. Example syntax for the relevant elements in a form template (.xsn) file is provided in [MS-IPFF] and [MS-IPFF2], and example syntax for a UDC file is provided in [MS-UDCX].

The following message is an example of a request message that a protocol client sends to the protocol server.

```
POST /_vti_bin/FormsServiceProxy.asmx HTTP/1.1
SOAPAction:
"http://schemas.microsoft.com/office/infopath/2007/formsServicesProxy/ForwardSoapRequest"
Content-Type: text/xml; charset="UTF-8"
User-Agent: SOAP Toolkit 3.0
Host: www.contoso.com
Content-Length: 805
Pragma: no-cache
Cookie: MSOWebPartPage_AnonymousAccessCookie=80; WSS_KeepSessionAuthenticated=80

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<tns:ForwardSoapRequest
xmlns:tns="http://schemas.microsoft.com/office/infopath/2007/formsServicesProxy"><tns:xmlCont
ent>
<targetNS:HelloWorld xmlns:targetNS="http://www.contoso.com/Service.asmx">
</targetNS:HelloWorld>
</tns:xmlContent>
<tns:version>1.0.0.3</tns:version>
<tns:url>http://www.contoso.com/FormServerTemplates/example.xsn</tns:url>
<tns:adapterName>query</tns:adapterName>
</tns:ForwardSoapRequest></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

The following message is an example of a successful response sent from the protocol server to the protocol client using the Forms Services Proxy Web Service Protocol.

```
HTTP/1.1 200 OK
Date: Mon, 21 Jan 2008 23:26:01 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Set-Cookie: WSS_KeepSessionAuthenticated=80; path=/
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Length: 561

<?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>
```

```
<ForwardSoapRequestResponse
xmlns="http://schemas.microsoft.com/office/infopath/2007/formsServicesProxy">
<ForwardSoapRequestResult>
<HelloWorldResponse xmlns="http://www.contoso.com/Service.asmx">
<HelloWorldResult>Hello World</HelloWorldResult>
</HelloWorldResponse>
</ForwardSoapRequestResult>
</ForwardSoapRequestResponse>
</soap:Body>
</soap:Envelope>
```

The following message is an example of a failure response sent from the protocol server to the protocol client using this protocol.

```
HTTP/1.1 500 Internal Server Error
Date: Mon, 21 Jan 2008 23:40:24 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Cache-Control: private
Content-Type: text/xml; charset=utf-8
Content-Length: 403

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:s="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<soap:Fault>
<faultcode>soap:Server</faultcode>
<faultstring>Server was unable to process request. ---&gt; Internal error.</faultstring>
<detail />
</soap:Fault>
</soap:Body>
</soap:Envelope>
```

The following message is an example of the request a protocol server sends the target Web service.

```
POST /anon/service1.asmx
HTTP/1.1
User-Agent: InfoPathDA
Content-Type: text/xml; charset="UTF-8"
SOAPAction: "http://www.contoso.com/Service/HelloWorld"
Host: www.contoso.com
Cache-Control: no-store,no-cache
Pragma: no-cache
Content-Length: 704
Expect: 100-continue
Connection: Keep-Alive

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
```

```
<wsse:UsernameToken>
<wsse:Username>CONTOSO\guest</wsse:Username>
</wsse:UsernameToken>
</wsse:Security>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
<targetNS:HelloWorld xmlns:targetNS="http://www.contoso.com/Service.asmx" />
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The following message is an example of the response the target Web service sends to the protocol server.

```
HTTP/1.1 200 OK
Date: Wed, 13 Feb 2008 19:35:16 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 1.1.4322
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Length: 374

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:s="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<HelloWorldResponse xmlns="http://www.contoso.com/Service.asmx">
<HelloWorldResult>Hello World</HelloWorldResult>
</HelloWorldResponse>
</soap:Body>
</soap:Envelope>
```

5 Security

5.1 Security Considerations for Implementers

In addition to the security considerations applicable to the underlying protocols, an implementation of the protocol server can mitigate risks by limiting the privileges of the identity which the protocol server uses for requests to the target Web service when no **authentication** element is present in the UDC file.

5.2 Index of Security Parameters

None.

6 Appendix A: Full WSDL

For ease of implementation the full WSDL follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://schemas.microsoft.com/office/infopath/2007/formsServicesProxy"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    targetNamespace="http://schemas.microsoft.com/office/infopath/2007/formsServicesProxy"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:types>
        <s:schema elementFormDefault="qualified"
targetNamespace="http://schemas.microsoft.com/office/infopath/2007/formsServicesProxy">
            <s:element name="ForwardSoapRequest">
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="1" maxOccurs="1" name="xmlContent">
                            <s:complexType mixed="true">
                                <s:sequence>
                                    <s:any />
                                </s:sequence>
                            </s:complexType>
                        </s:element>
                        <s:element minOccurs="1" maxOccurs="1" name="version"
                                  type="s:string" />
                        <s:element minOccurs="1" maxOccurs="1" name="url"
                                  type="s:string" />
                        <s:element minOccurs="1" maxOccurs="1" name="adapterName"
                                  type="s:string" />
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="ForwardSoapRequestResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="1" maxOccurs="1"
                                  name="ForwardSoapRequestResult">
                            <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="ForwardSoapRequestSoapIn">
        <wsdl:part name="parameters" element="tns:ForwardSoapRequest" />
    </wsdl:message>
    <wsdl:message name="ForwardSoapRequestSoapOut">
        <wsdl:part name="parameters" element="tns:ForwardSoapRequestResponse" />
    </wsdl:message>
    <wsdl:portType name="WebServiceProxySoap">
        <wsdl:operation name="ForwardSoapRequest">
```

```
            <wsdl:input message="tns:ForwardSoapRequestSoapIn" />
            <wsdl:output message="tns:ForwardSoapRequestSoapOut" />
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="WebServiceProxySoap" type="tns:WebServiceProxySoap">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="ForwardSoapRequest">
            <soap:operation
soapAction="http://schemas.microsoft.com/office/infopath/2007/formsServicesProxy/ForwardSoapR
equest" 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="WebServiceProxySoap12" type="tns:WebServiceProxySoap">
        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="ForwardSoapRequest">
            <soap12:operation
soapAction="http://schemas.microsoft.com/office/infopath/2007/formsServicesProxy/ForwardSoapR
equest" 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 Forms Server 2007
- Microsoft® Office InfoPath® 2007
- Microsoft® InfoPath® 2010
- Microsoft® Office SharePoint® Server 2007
- Microsoft® SharePoint® Server 2010

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: This protocol implementation returns a SOAP fault if the form template (.xsn) file identified by the **url** and **version** parameters is not a **browser-enabled form template**.

<2> Section 3.1.4.1: This protocol implementation will fail and respond with a SOAP fault if the target Web service is itself an implementation of this protocol.

<3> Section 3.1.4.1: Office SharePoint Server 2007, Office Forms Server 2007 and SharePoint Server 2010 will always include a username element, and will use an empty string for this element's value if the user of the protocol client is not authenticated by the protocol server.

8 Change Tracking

This section identifies changes that were made to the [MS-FSPP] protocol document between the March 2011 and June 2011 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 ForwardSoapRequest	Added message processing descriptions from "ForwardSoapRequestSoapIn" and "ForwardSoapRequestSoapOut".	N	Content updated.
3.1.4.1.1.1 ForwardSoapRequestSoapIn	Removed message processing description.	N	Content updated.
3.1.4.1.1.2 ForwardSoapRequestSoapOut	Removed message processing description.	N	Content updated.

9 Index

T

V

W