[MS-FSFDP]:

Forms Services Feature Detection 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

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

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	Minor	Updated 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	Editorial	Changed language and formatting in the technical content.
7/23/2010	2.05	Minor	Clarified the meaning of the technical content.
9/27/2010	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.
3/18/2011	2.05	No Change	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	2.6	Minor	Clarified the meaning 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	3.0	No Change	No changes to the meaning, language, or formatting of the technical content.
9/12/2012	3.0	No Change	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	3.0.1	Editorial	Changed language and formatting in the technical content.
2/11/2013	3.0.1	No Change	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	4.0	Major	Significantly changed the technical content.
11/18/2013	4.0	No Change	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	4.0	No Change	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
4/30/2014	4.0	No Change	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	4.0	No Change	No changes to the meaning, language, or formatting of the technical content.
8/24/2015	5.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The Forms Services Feature Detection Protocol enables a protocol client to detect status of **form server** features, get the form server version, and get the constructed URL that is required to render a form in a Web browser.

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:

200 OK: A response to indicate that the request has succeeded.

absolute URI: An absolute **Uniform Resource Identifier (URI)**, as described in [RFC3986].

ASCII: The American Standard Code for Information Interchange (ASCII) is an 8-bit character-encoding scheme based on the English alphabet. ASCII codes represent text in computers, communications equipment, and other devices that work with text. ASCII refers to a single 8-bit ASCII character or an array of 8-bit ASCII characters with the high bit of each character set to zero.

Augmented Backus-Naur Form (ABNF): A modified version of Backus-Naur Form (BNF), commonly used by Internet specifications. ABNF notation balances compactness and simplicity with reasonable representational power. ABNF differs from standard BNF in its definitions and uses of naming rules, repetition, alternatives, order-independence, and value ranges. For more information, see [RFC5234].

authentication: The act of proving an identity to a server while providing key material that binds the identity to subsequent communications.

feature: A package of SharePoint elements that can be activated or deactivated for a specific feature scope.

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.

form file: An XML file that contains data that is entered into an InfoPath form by using a web browser or Microsoft InfoPath.

form server: A server that can host XML-based electronic forms and that supports rendering those forms in a web browser.

form template: A file or set of files that defines the data structure, appearance, and behavior of a **form**.

form template (.xsn) file: A cabinet (.cab) file with an .xsn file name extension that contains the files that comprise a form template.

HTTP method: In an HTTP message, a token that specifies the method to be performed on the resource that is identified by the **Request-URI**, as described in [RFC2616].

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 Secure (HTTPS): An extension of HTTP that securely encrypts and decrypts web page requests. In some older protocols, "Hypertext Transfer Protocol over Secure Sockets Layer" is still used (Secure Sockets Layer has been deprecated). For more information, see [SSL3] and [RFC5246].

message body: The content within an HTTP message, as described in [RFC2616] section 4.3.

query component: A portion of a **URL** that follows a question mark (?), as described in [RFC3986].

rendering URL: The URL that is used to render an InfoPath form in a web browser if the form cannot be opened by using Microsoft InfoPath.

Request-URI: A **URI** in an **HTTP** request message, as described in [RFC2616].

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.

site collection: A set of websites (1) that are in the same content database, have the same owner, and share administration settings. A site collection can be identified by a GUID or the **URL** of the top-level site for the site collection. Each site collection contains a top-level site, can contain one or more subsites, and can have a shared navigational structure.

Status-Code: A 3-digit integer result code in an HTTP response message, as described in [RFC2616].

Status-Line: The first line of an HTTP response message, as described in [RFC2616].

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The **Unicode** standard [UNICODE5.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

Uniform Resource Identifier (URI): A string that identifies a resource. The URI is an addressing mechanism defined in Internet Engineering Task Force (IETF) Uniform Resource Identifier (URI): Generic Syntax [RFC3986].

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].

UTF-8: A byte-oriented standard for encoding Unicode characters, defined in the Unicode standard. Unless specified otherwise, this term refers to the UTF-8 encoding form specified in [UNICODE5.0.0/2007] section 3.9.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[RFC1945] Berners-Lee, T., Fielding, R., and Frystyk, H., "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996, http://www.ietf.org/rfc/rfc1945.txt

[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

[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

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.rfc-editor.org/rfc/rfc5234.txt

1.2.2 Informative References

[MS-FSDAP] Microsoft Corporation, "Forms Services Design and Activation Web Service Protocol".

1.3 Overview

This protocol enables a protocol client to communicate with a protocol server over a **Hypertext Transfer Protocol (HTTP)** connection to perform the following supported functions:

- **Form Server Detection:** Using this protocol function, the protocol client can detect if form server **features** are present and enabled on the protocol server. The protocol client sends an HTTP request to the protocol server with a parameter to detect whether form server features are enabled, and the protocol server sends back an HTTP response containing the result.
- **Form Server Version Retrieval:** Using this protocol function, the protocol client can get the form server version after detecting if form server features are present and enabled on the protocol server. The protocol client sends an HTTP request to the protocol server with a parameter to get the form server version, and the protocol server sends back an HTTP response containing the result.
- **Rendering URL Construction:** Using this protocol function, the protocol client can construct the **URL** that is required to render a new or existing **form** in a Web browser. The protocol client sends an HTTP request to the protocol server, and the protocol server sends back an HTTP response containing a URL that can be used to render the form in a Web browser.

1.4 Relationship to Other Protocols

For message transport, this protocol uses the HTTP/1.0 protocol, as described in [RFC1945], the HTTP/1.1 protocol, as described in [RFC2616], or the **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)** protocol, as described in [RFC2818].

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

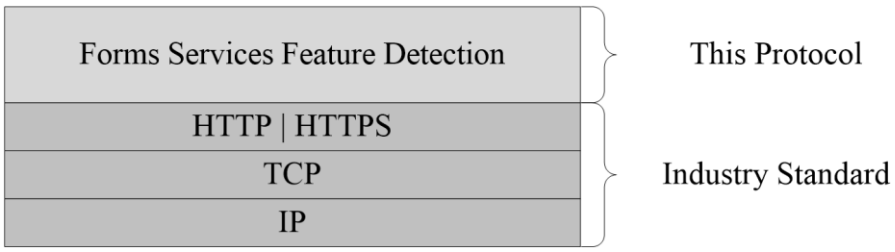

Figure 1: This protocol in relation to other protocols

This protocol is intended to be used as a prerequisite to calling the Forms Services Design and Activation Web Service Protocol, as specified in [MS-FSDAP].

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 "/_layouts/FormServerDetector.aspx"or "/_layouts/15/FormServerDetector.aspx" to the URL of the site.

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

1.6 Applicability Statement

The protocol client can use this protocol in the following scenarios:

- **Form Server Detection:** This function can be used by the protocol client to detect if the protocol server supports form server functionality.
- **Form Server Version Retrieval:** This function can be used by the protocol client to get the form server version.
- **Rendering URL Construction:** This function can be used to construct the required URL for rendering a form in a Web browser if the form cannot be opened by the protocol client.

1.7 Versioning and Capability Negotiation

This protocol uses multiple transports with either HTTP or HTTPS, as described in section 2.1.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Protocol servers MUST support HTTP, as specified in [RFC1945] or [RFC2616]. Protocol servers SHOULD additionally support HTTPS, as specified in [RFC2818], for securing communication with clients.

2.2 Message Syntax

The following subsections specify the different parts of HTTP request and response messages.

2.2.1 Request Syntax

2.2.1.1 Request HTTP Version

The HTTP version as specified in [RFC1945] section 3.1 or [RFC2616] section 3.1 MUST be either "HTTP/1.0" or "HTTP/1.1" for the requests that use this protocol.

2.2.1.2 Request HTTP Method

The **HTTP method** specified in [RFC1945] section 8 or [RFC2616] section 9 MUST be GET, as specified in [RFC1945] section 8.1 or [RFC2616] section 9.3, for the requests that use this protocol.

2.2.1.3 Request-URI Syntax

2.2.1.3.1 Request-URI Details

The **Request-URI** MUST be a valid **Uniform Resource Identifier (URI)**, as specified in [RFC3986].

The following **Augmented Backus-Naur Form (ABNF)**, as specified in [RFC5234], specifies the syntax of the **Request-URI**.

```
Request-URI     = (path) "?" (query-component)
path            = (base) "/_layouts" (opt_ver) "/FormServerDetector.aspx"
opt ver         = "" / "/15"
base            = scheme ":" hier-part
```

The **base** ABNF rule identifies the site on the server to which the request was made. The ABNF for **scheme** and **hier-part** is specified in [RFC3986] section 3.3. The **query component** in the **Request-URI** is specified in the following section. The form server SHOULD support the value of "/15" **opt_ver**<1> but MAY respond with HTTP **Status-Code** 404<2>.

2.2.1.3.2 Query Component Details

The query component of the **Request-URI** MUST be present. The following subsections specify the syntax of the query component for the three functions that are supported by this protocol.

The following ABNF, as defined in [RFC5234], specifies the syntax that the query component MUST adhere to.

```
query-component = (query-detection / query-version / query-URL)
```

The protocol server MUST interpret the value of the query component as case-insensitive.

2.2.1.3.2.1 Request for Form Server Detection

To use the form server detection function, the query component of the **Request-URI** MUST have the *IsFormServerEnabled* query parameter, which MUST be immediately followed by "=".

The following ABNF specifies the syntax that the query detection MUST adhere to.

```
query-detection = "IsFormServerEnabled=" query
```

The ABNF for the **query** rule is specified in [RFC3986] section 3.

If the protocol client passes any query parameters in addition to the *IsFormServerEnabled* parameter, the protocol server MUST ignore these additional parameters and the request for form server detection MUST take precedence.

2.2.1.3.2.2 Request for Form Server Version Retrieval

The protocol server SHOULD<3> support the form server version retrieval function. To use this method, the query component of the **Request-URI** MUST have the *FormServerVersion* parameter, which MUST be immediately followed by "=".

The following ABNF specifies the syntax that the query version MUST adhere to.

```
query-version = "FormServerVersion=" query
```

The ABNF for the **query** rule is specified in [RFC3986] section 3.

If the *FormServerVersion* query parameter is not supported by the protocol server, this query parameter MUST be ignored. If no other supported query parameters exist in the query component, the protocol server MUST return Status-Code 204, as specified in section 2.2.2.1.1.

If the protocol client passes the *IsFormServerEnabled* parameter, the request for form server detection MUST take precedence.

If the *FormServerVersion* parameter is supported by the protocol server and the protocol client passes additional parameters other than the *IsFormServerEnabled* parameter, the protocol server MUST ignore these additional parameters.

2.2.1.3.2.3 Request for Rendering URL Construction

To use the **rendering URL** construction function, the query component and its query URL MUST adhere to the syntax that is specified in the following ABNF.

```
query-URL              = (xmlLocation-parameter / xsnLocation-parameter)
   ["&" saveLocation-parameter]
xmlLocation-parameter  = "XmlLocation=" (value)
xsnLocation-parameter  = "XsnLocation=" (value)
saveLocation-parameter = "SaveLocation=" (value)
value                  = (scheme ":" hier-part) /
   (["~site" / "~site-collection"] path-absolute)
```

The ABNF for the **scheme**, **path-absolute** and **hier-part** rules is specified in [RFC3986] section 3.

As specified in the preceding ABNF, the query component supports three parameters:

- *XmlLocation*
- *XsnLocation*
- *SaveLocation*

The following table specifies the meaning of these parameters.

Parameter	Description
XmlLocation	The path to a **form file** on the protocol server. MUST be an **ASCII** **string** that specifies the location of the form file that needs to be rendered on the protocol server. MUST follow the format specified in [RFC3986].
XsnLocation	The path to a **form template (.xsn) file** on the protocol server. MUST be an ASCII **string** that specifies the location of the **form template**, which can be used to generate a form file to be rendered on the protocol server. MUST follow the format specified in [RFC3986].
SaveLocation	The path to a folder on the protocol server in the same **site collection** as FormServerDetector.aspx. MUST be an ASCII **string** that specifies the location where the form file can be saved, if needed. MUST follow the format specified in [RFC3986].

The values of the parameters in the query component MUST NOT contain any un-escaped characters that are listed as "reserved" in [RFC3986] section 2.2.

To construct the rendering URL for an existing form, the *XmlLocation* parameter MUST be specified. To construct the rendering URL for a new form, the *XsnLocation* parameter MUST be specified. In both cases, the *SaveLocation* parameter is optional.

Any other combination of the supported parameters MUST be treated as input that is not valid, and in such a case, the protocol server MUST return Status-Code 204, as specified in section 2.2.2.1.1. Any additional parameters, other than those specified in the preceding table, MUST be ignored by the protocol server.

The protocol server MUST NOT require that the parameters appear in a particular order.

2.2.1.4 Request Headers Syntax

The following request header is relevant to this protocol:

- **Accept:** This header is specified in [RFC1945] section D.2.1 or [RFC2616] section 14.1. The protocol client SHOULD specify this header with the value "*/*". The protocol server SHOULD ignore the value of this header.

2.2.2 Response Syntax

2.2.2.1 Response Status-Line

The response **Status-Line** MUST be valid according to [RFC1945] section 6.1 or [RFC2616] section 6.1.

2.2.2.1.1 Success Response

The protocol server MUST return HTTP Status-Code **200 OK** to indicate a success response, as specified in section 3.2.5. The response body MUST contain detailed results, as specified in section 2.2.2.3.

For success responses other than those specified for Status-Code 200 OK, the protocol server MUST return HTTP Status-Code 204 No Content as specified in section 3.2.5, and in [RFC1945] section 9.2 or [RFC2616] section 10.2.5.

2.2.2.1.2 Failure Response

The protocol server MUST return an HTTP 4xx or 5xx Status-Code, as specified in [RFC1945] section 6.1.1 or [RFC2616] section 6.1.1, to indicate that the request failed.

The protocol server SHOULD return the HTTP Status-Code 401 to indicate that the protocol client can retry the request using a different authentication protocol or properties, but MAY<4> return a different code for this condition.

2.2.2.2 Response Headers

The following response headers are relevant to this protocol:

- **Content-Length:** Specified in [RFC1945] section 10.4 or [RFC2616] section 14.13.
- **Content-Type:** Specified in [RFC1945] section 10.5 or [RFC2616] section 14.17. MUST be present and MUST be set to "text/html; charset=utf-8" for Status-Code 200 OK.

2.2.2.3 Response Body Syntax

The response body returned from the protocol server for the functions that are supported by this protocol is specified in the following subsections. Failure responses for all functions MUST return a response with the **Content-Length** header set to zero and with no **message body**. The protocol client MUST interpret the response as case-sensitive.

2.2.2.3.1 Response for Form Server Detection Request

HTTP 200 OK responses MUST return a message body of **UTF-8** encoded text, as defined in [RFC5234] and specified in the following ABNF.

```
message-body = "<server" %x20 "IsFormServerEnabled" %x20 "=" %x20 "'true'"   %x20 "/>"
```

All white spaces MUST be preserved and any additional white spaces MUST NOT be added.

2.2.2.3.2 Response for Form Server Version Retrieval Request

If the form server supports this protocol function, HTTP 200 OK responses MUST return a message body of UTF-8 encoded text, as defined in [RFC5234] and specified in the following ABNF.

```
message-body =/ "<server" %x20 "FormServerVersion" %x20 "=" %x20 "'15'" %x20 "/>"
```

All white spaces MUST be preserved and any additional white spaces MUST NOT be added.

If the form server does not support this protocol function, the protocol server MUST NOT return a response body, but instead, it MUST return Status-Code 204, as specified in section 2.2.2.1.1.

2.2.2.3.3 Response for Rendering URL Construction Request

Responses with Status-Code 200 OK MUST return a message body, as specified in the following ABNF.

```
message-body   =/ "OpenInFormServer=" (rendering-url)
rendering-url  = (url-path) "?" (query-component) "&OpenIn=Browser"
url-path       = scheme ":" hier-part "/_layouts/15/FormServer.aspx"
```

The **rendering-url** rule in the ABNF refers to the URL that the protocol server MUST return so that the form can be rendered in a Web browser. It MUST be a valid **absolute URI**, as specified in [RFC3986] section 4.3. If the form server is enabled on the protocol server, FormServer.aspx MUST exist in the site collection.

The **query-component** rule in the constructed URL MUST contain only the supported parameters that are sent from the protocol client, as specified in section 2.2.1.3.2.3. The protocol server SHOULD return un-escaped characters when the value of the query parameter contains **Unicode** characters, but MAY return percent-encoded characters, as specified in [RFC3986] section 2.1, for the query parameter values in the query component. When the *SaveLocation* parameter is present in **rendering-url**, the protocol server SHOULD replace the "&" character before the *SaveLocation* parameter with a "?" character in the constructed rendering URL.

The ABNF for **scheme** and **hier-part** is specified in [RFC3986] section 3.3.

3 Protocol Details

3.1 Common Details

This section specifies details common to both protocol server and protocol client behavior.

Except where specified, protocol clients SHOULD interpret HTTP Status-Codes returned by the protocol server as specified in [RFC1945] section 9 or [RFC2616] section 10.

This protocol allows protocol servers to perform implementation-specific authorization checks and notify protocol clients of authorization faults using HTTP Status-Codes.

3.1.1 Abstract Data Model

This section specifies a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The specified 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 specified in this document.

The following paragraphs specify the following terms in the context of this protocol.

Base URL: The portion of the **Request-URI** that matches the **base** rule in the ABNF in section 2.2.1.3.1.

Query Parameters: The parameters in the query component of the **Request-URI**, as specified in section 2.2.1.3.2, and its subsections.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

3.2.1 Abstract Data Model

As specified in section 3.1.1.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

The Request-URI that the protocol client sends to the protocol server MUST contain the query component and MUST follow the rules that are specified in section 2.2.1.3.

The protocol client MUST interpret the response based on the HTTP Status-Code as follows:

- **Status-Code** 200 OK: The request was successful and form server features are enabled on the protocol server. The response MUST be interpreted as specified in the following table.

Protocol Function	Status-Code	Meaning of Status-Code
Form Server Detection	200	Form server features are enabled on the protocol server. The response body contains the text specified in section 2.2.2.3.1.
Form Server Version Retrieval	200	Form server features are enabled on the protocol server and the form server version was successfully returned. The response body contains the text specified in section 2.2.2.3.2.
Rendering URL Construction	200	Rendering URL has been successfully constructed. The response body contains the rendering URL in the format specified in section 2.2.2.3.3.

- **Status-Code 204:** The request was successful, and the response MUST be interpreted as specified in the following table. The protocol server MUST NOT return a response body.

Protocol Function	Status-Code	Meaning of Status-Code
Form Server Detection	204	Form server features are not enabled on the protocol server.

Protocol Function	Status-Code	Meaning of Status-Code
Form Server Version Retrieval	204	The *FormServerVersion* query parameter is not supported by the protocol server or form server features are not enabled on the protocol server, and the form server version was not returned.
Rendering URL Construction	204	The protocol server cannot construct the rendering URL, based on the given parameters. Possible reasons can be incorrect syntax or value in the query parameter, or form server features are not enabled on the protocol server.

- **Status-Code 302:** As specified in [RFC2616] section 10.3.3.
- **Status-Code 4xx/5xx:** The request failed. The response body MUST NOT contain the text specified in section 2.2.2.3.1, section 2.2.2.3.2, and section 2.2.2.3.3, but can include informative text providing details of the failure.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Server Details

3.3.1 Abstract Data Model

As specified in section 3.1.1.

3.3.2 Timers

None.

3.3.3 Initialization

None.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

The protocol server MUST process request messages received from a protocol client as follows:

- the protocol server MUST validate that the request syntax matches the syntax specified in section 2.2.1.3. If the syntax is not valid, the protocol server MUST return Status-Code 204.

- before checking any of the other query parameters in the query component, the protocol server MUST look for the presence of the *IsFormServerEnabled* parameter. If the parameter *IsFormServerEnabled* exists:
  - the protocol server MUST process the request as a form server detection request and MUST ignore any extra parameters.
  - if the form server is enabled on the protocol server, the protocol server MUST return Status-Code 200 OK. For such a response, the protocol server MUST generate a response body that MUST contain the text specified in section 2.2.2.3.1.
  - if the form server is not enabled on the protocol server, the protocol server MUST return Status-Code 204 No Content.
- if the parameter *IsFormServerEnabled* does not exist in the query parameters, the protocol server SHOULD<5> look for the presence of the *FormServerVersion* parameter. If the parameter *FormServerVersion* exists and is supported by the protocol server:
  - the protocol server MUST process the request as a form server version retrieval request and MUST ignore any extra parameters.
  - if the form server is enabled on the protocol server, the protocol server MUST return Status-Code 200 OK. For such a response, the protocol server MUST generate a response body that MUST contain the text specified in section 2.2.2.3.2.
  - if the form server is not enabled on the protocol server, the protocol server MUST return Status-Code 204 No Content.
- if the parameter *IsFormServerEnabled* does not exist in the query parameters, and either the form server version retrieval method is not supported or the *FormServerVersion* query parameter is not present:
  - the protocol server MUST verify that the parameter list conforms to one of the supported combinations specified in section 2.2.1.3.2.3. If the combination of parameters is not valid, the protocol server MUST return Status-Code 204 No Content.
  - the protocol server MUST validate that the values for the *XmlLocation* and *XsnLocation* parameters are valid, as specified in the table in section 2.2.1.3.2.3. If the validation fails, the protocol server MUST return Status-Code 204 No Content.
  - the protocol server MUST validate that the protocol client has permission to the resources specified by the parameter values. If the validation fails, the protocol server SHOULD return Status-Code 401 Unauthorized, but MAY<6> return a different code.
  - if the validation succeeds and form server features are enabled on the protocol server, the protocol server MUST return Status-Code 200 OK. For a response with Status-Code 200 OK, the protocol server MUST generate a response body as specified in section 2.2.2.3.3, and as shown in the example in section 4.3.2.1.
  - if form server features are not enabled on the protocol server, the protocol server MUST return Status-Code 204 No Content.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

4 Protocol Examples

This section illustrates the messages exchanged when a protocol client makes a successful HTTP request to a protocol server using this protocol.

4.1 Form Server Detection

The example in the following subsections shows the client and server interaction during form server detection.

4.1.1 Client Request

The following example is a protocol client request to detect whether form server features are enabled on the protocol server.

```
GET / layouts/FormServerDetector.aspx?IsFormServerEnabled=check HTTP/1.1
Accept: */*
Host: www.contoso.com
```

4.1.2 Server Response

4.1.2.1 Response When Form Server Is Enabled

The following example shows the response text when form server features are enabled on the protocol server.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 39
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET

<server IsFormServerEnabled = 'true' />
```

4.1.2.2 Response When Form Server Is Not Enabled

The following example shows the response text when form server features are not enabled on the protocol server.

```
HTTP/1.1 204 No Content
Cache-Control: Private
Content-Length: 0
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
```

4.2 Form Server Version Retrieval

The example in the following subsections shows client and server interaction during the form server version retrieval.

4.2.1 Client Request

The following example is a protocol client request to get the version of form server.

```
GET /_layouts/FormServerDetector.aspx?FormServerVersion=check HTTP/1.1
Accept: */*
Host: www.contoso.com
```

4.2.2 Server Response

4.2.2.1 Response When Form Server Is Enabled

The following example shows the response text when form server features are enabled on the protocol server.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 35
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET

<server FormServerVersion = '15' />
```

4.2.2.2 Response When Form Server Is Not Enabled

The following example shows the response text when form server features are not enabled on the protocol server.

```
HTTP/1.1 204 No Content
Cache-Control: Private
Content-Length: 0
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
```

4.3 Rendering URL Construction

The example in the following subsections shows client and server interaction during the construction of the rendering URL.

4.3.1 Client Request

The following example is a request to obtain the rendering URL of an existing form from the protocol server.

```
GET /_layouts/FormServerDetector.aspx?XmlLocation=/Folder/filename.xml HTTP/1.1
Accept: */*
Host: www.contoso.com
```

4.3.2 Server Response

4.3.2.1 Response When Form Server Is Enabled

The following example shows the response text of a request as shown in section 4.3.1, when the parameter is valid and form server features are enabled on the protocol server.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 102
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET

OpenInFormServer=http://www.contoso.com/_layouts/FormServer.aspx?XmlLocation=/Folder/filename
.xml&OpenIn=Browser
```

4.3.2.2 Response When Form Server Is Not Enabled

The following example shows the response text of a request as given in section 4.3.1, when form server features are not enabled on the protocol server.

```
HTTP/1.1 204 No Content
Cache-Control: Private
Content-Length: 0
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
```

5 Security

5.1 Security Considerations for Implementers

This protocol introduces no additional security considerations beyond those applicable to its underlying protocols.

5.2 Index of Security Parameters

None.

6 Appendix A: 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 InfoPath 2013
- Microsoft Office SharePoint Server 2007
- Microsoft SharePoint Server 2010
- Microsoft SharePoint Server 2013
- Microsoft SharePoint Server 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 2.2.1.3.1: SharePoint Server 2010 exposes this protocol by appending "/_layouts/FormServerDetector.aspx" to the URL of the site; SharePoint Server 2013 exposes this protocol by appending "/_layouts/15/FormServerDetector.aspx" to the URL of the site.

<2> Section 2.2.1.3.1: Microsoft SharePoint Server 2010 returns HTTP Status-Code 404

<3> Section 2.2.1.3.2.2: The form server version retrieval function is only supported in SharePoint Server 2010.

<4> Section 2.2.2.1.2: Office SharePoint Server 2007 and SharePoint Server 2010 return a 401 Unauthorized Status-Code if the client is not authorized to access the path in the **Request-URI**. If the client is authorized to access the path in the **Request-URI** but is not authorized to access a resource identified by the query parameters in the **Request-URI**, SharePoint Server 2010 returns a 204 No Content Status-Code and Office SharePoint Server 2007 returns a 302 Found Status-Code.

<5> Section 3.3.5: The *FormServerVersion* query parameter is only supported in SharePoint Server 2010.

<6> Section 3.3.5: If the client is authorized to access the path in the **Request-URI** but is not authorized to access a resource identified by the query parameters in the **Request-URI**, SharePoint Server 2010 returns a 204 No Content Status-Code and Office SharePoint Server 2007 returns a 302 Found Status-Code.

7 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**.

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
6 Appendix A: Product Behavior	Updated list of supported products.	Y	Content updated due to protocol revision.

8 Index