[MS-IPDSP]:

InfoPath Digital Signing Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **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 can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might 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, email addresses, logos, people, places, and events that are 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 as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does 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 documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
2/19/2010	1.0	Major	Initial Availability
3/31/2010	1.01	Editorial	Revised and edited the technical content
4/30/2010	1.02	Editorial	Revised and edited the technical content
6/7/2010	1.03	Editorial	Revised and edited the technical content
6/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
7/23/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	1.04	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	1.04	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	2.0	Major	Significantly changed the technical content.
4/11/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
9/12/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	2.0.1	Editorial	Changed language and formatting in the technical content.
2/11/2013	2.0.1	None	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	2.1	Minor	Clarified the meaning of the technical content.
11/18/2013	2.1	None	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	2.1	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	2.1	None	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	2.1	None	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	2.1	None	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
2/26/2016	3.0	Major	Significantly changed the technical content.
7/15/2016	3.0	None	No changes to the meaning, language, or formatting of the technical content.
9/14/2016	3.0	None	No changes to the meaning, language, or formatting of the technical content.
7/24/2018	4.0	Major	Significantly changed the technical content.
10/1/2018	5.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The InfoPath Digital Signing Protocol specifies a mechanism by which a protocol client can work with a protocol server to apply a digital signature to a form file.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

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.

base64 encoding: A binary-to-text encoding scheme whereby an arbitrary sequence of bytes is converted to a sequence of printable ASCII characters, as described in [RFC4648].

certificate: A certificate is a collection of attributes and extensions that can be stored persistently. The set of attributes in a certificate can vary depending on the intended usage of the certificate. A certificate securely binds a public key to the entity that holds the corresponding private key. A certificate is commonly used for **authentication** and secure exchange of information on open networks, such as the Internet, extranets, and intranets. Certificates are digitally signed by the issuing certification authority (CA) and can be issued for a user, a computer, or a service. The most widely accepted format for certificates is defined by the ITU-T X.509 version 3 international standards. For more information about attributes and extensions, see [RFC3280] and [X509] sections 7 and 8.

ciphertext: The encrypted form of a message. Ciphertext is achieved by encrypting the plaintext form of a message, and can be transformed back to plaintext by decrypting it with the proper key. Without that transformation, a ciphertext contains no distinguishable information.

digest: The fixed-length output string from a one-way hash function that takes a variable-length input string and is probabilistically unique for every different input string. Also, a cryptographic checksum of a data (octet) stream.

digital signature: A value that is generated by using a digital signature algorithm, taking as input a private key and an arbitrary-length string, such that a specific verification algorithm is satisfied by the value, the input string, and the public key corresponding to the input private key.

empty string: A string object or variable that is initialized with the value "".

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

form view: A display setting that is saved with an InfoPath form template and specifies which controls and data appear on a form when the form is being filled out.

HTTP OK: An **HTTP** response with status code 200, as described in [RFC2616] section 6.1.1.

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 1.0 (HTTP/1.0): An application of the Standard Generalized Markup Language (SGML) that uses tags to mark elements in a document, as described in [HTML].

Hypertext Transfer Protocol 1.1 (HTTP/1.1): Version 1.1 of the Hypertext Transfer Protocol (HTTP), as described in [RFC2068].

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.

persist: The process of storing data in a memory medium that does not require electricity to maintain the data that it stores. Examples of such mediums are hard disks, CDs, non-volatile RAM, and memory sticks.

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.

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

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].

XML: The Extensible Markup Language, as described in [XML1.0].

XML document: A document object that is well formed, as described in [XML10/5], and might be valid. An XML document has a logical structure that is composed of declarations, elements, comments, character references, and processing instructions. It also has a physical structure that is composed of entities, starting with the root, or document, entity.

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.

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.

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

[MS-IPFFX] Microsoft Corporation, "InfoPath Form File Format".

[MS-IPFF] Microsoft Corporation, "InfoPath Form Template 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.rfc-editor.org/rfc/rfc2616.txt

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006, http://www.rfc-editor.org/rfc/rfc4648.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

[W3C-XML] Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F., Eds., "Extensible Markup Language (XML) 1.1 (Second Edition)", W3C Recommendation, August 2006, http://www.w3.org/TR/2006/REC-xml11-20060816/

[XMLDSig] Bartel, M., Boyer, J., Fox, B., et al., "XML-Signature Syntax and Processing", W3C Recommendation, February 2002, http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/

1.2.2 Informative References

[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

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

1.3 Overview

This protocol enables a protocol client to communicate with a protocol server over a **Hypertext Transfer Protocol (HTTP)** connection to apply a **digital signature** to a **form file** that is stored on the protocol server. To apply a digital signature to a form file stored on the protocol server, the protocol client performs the following supported functions:

- **Retrieve Form File Hash**: The protocol client sends this message to the protocol server to initiate the application of a digital signature to a form file. Using this protocol function, the protocol client sends a request to the protocol server that contains local information, including a rendered image of the form file and operating system information, to be embedded in the signed form file. The protocol server sends back an HTTP response containing a **digest** of the local information sent by the protocol client, and other data stored in the form file. See section 5.1 for security considerations.

- **Add Signature Value and Context**: Using this protocol function, the protocol client generates encrypted **ciphertext** of the digest value returned in the HTTP response to the **Retrieve Form File Hash** request message. The ciphertext value, and the **certificate** used to encrypt it, are sent to the protocol server and stored in the form file as **value** and **context**, respectively, of the digital signature. The application of the digital signature is complete when the protocol server successfully processes this request. See section 5.1 for security considerations.
- **Cancel Digital Signature**: Using this protocol function, the protocol client sends an HTTP request to the protocol server to notify the protocol server that the signing process has been cancelled.

1.4 Relationship to Other Protocols

This protocol requires the **Hypertext Transfer Protocol 1.0 (HTTP/1.0)**, as described in [RFC1945], or the **Hypertext Transfer Protocol 1.1 (HTTP/1.1)**, as described in [RFC2616], protocol for message transport. 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:

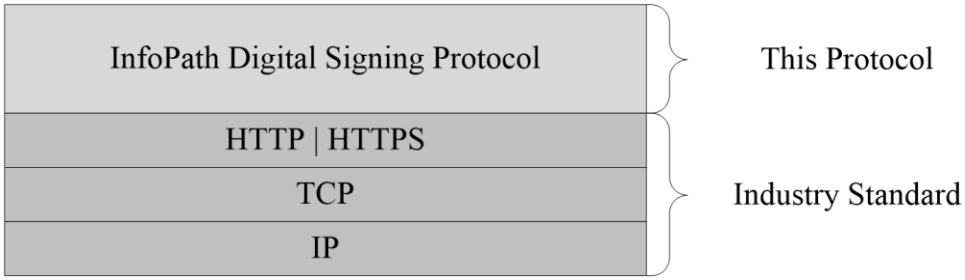

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

Prerequisites and preconditions of **HTTP**, as described in [RFC2616], apply to this protocol.

- This protocol operates against a **site** identified by a **URL** known by the protocol client.
- The data represented by the **Protocol Server Specific Event Log**, as described in section 2.2.1.2.1.1, are known by the protocol client.
- **DataDefinition**, as described in section 2.2.1.2.1, is known by the protocol client.
- **PageStateData**, as described in section 2.2.2.2.1.3, is known by the protocol client.

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

1.6 Applicability Statement

The operations described by this protocol apply to a protocol client that interacts with a protocol server to create and apply a **digital signature** to a **form file** stored on the protocol server. This protocol is intended for use by protocol clients and protocol servers connected by high-bandwidth and low-latency network connections.

1.7 Versioning and Capability Negotiation

Supported Transports: This protocol uses multiple transports with **HTTP**, as described in section 2.1.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

All protocol messages MUST be transmitted in the **HTTP** protocol syntax specified in [RFC2616]. Protocol servers MUST support the HTTP protocol. Protocol servers SHOULD additionally support **HTTPS** for secure communication with protocol clients.

2.2 Message Syntax

This section specifies the message syntax and details of the protocol messages transported between the protocol client and the protocol server.

2.2.1 Request Syntax

2.2.1.1 Request Metadata

The protocol client MUST send protocol messages to the protocol server using **HTTP** **POST**, as specified in [RFC2616] section 9.5. The protocol message **Request-URI** MUST be a valid **URI**, as specified in [RFC3986].

2.2.1.2 Request Entity Body details

The following subsections specify the syntax of the request entity body generated by the three functions supported by this protocol.

2.2.1.2.1 Request for Retrieve Form File Hash

This message is sent by the protocol client to initiate the signing process.

The **message body** MUST be an **XML document**, as specified in [W3C-XML], which conforms to the following **XML schema**:

```
<xsd:element name="SignRequest">
<xsd:complexType>
        <xsd:all>
            <xsd:element ref="EventLog" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="DataDefinition" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="Comment" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="Time" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="OS" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="Browser" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="Version" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="Monitors" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="Width" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="Height" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="Colors" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="PNG" minOccurs="1" maxOccurs="1"/>
        </xsd:all>
    </xsd:complexType>
</xsd:element>

<xsd:element name="EventLog" type="xsd:string"/>
<xsd:element name="DataDefinition" type="xsd:string"/>
<xsd:element name="Comment" type="xsd:string"/>
<xsd:element name="Time" type="xsd:string"/>
<xsd:element name="OS" type="xsd:string"/>
<xsd:element name="Browser" type="xsd:string"/>
<xsd:element name="Version" type="xsd:string"/>
```

```
<xsd:element name="Monitors" type="xsd:string"/>
<xsd:element name="Width" type="xsd:string"/>
<xsd:element name="Height" type="xsd:string"/>
<xsd:element name="Colors" type="xsd:string"/>
<xsd:element name="PNG" type="xsd:base64Binary"/>
```

EventLog: As specified in section 2.2.1.2.1.1.
DataDefinition: The **string** indicating the signed data block of the **form file** being signed, as specified in [MS-IPFF] section 2.2.15 and [MS-IPFF2] section 2.2.1.1.15.
Comment: The comment for the **digital signature** specified in [MS-IPFFX] section 2.1.2.1.
Time: The system date and time for the client computer specified in [MS-IPFFX] section 2.1.2.3.
OS: The version of the operating system running on the client computer at the time of signing, as specified in [MS-IPFFX] section 2.1.2.5.
Browser: The name and the version of the Web browser used on the client computer to sign the form, as specified in [MS-IPFFX] section 2.1.2.9.
Version: The version of the protocol server that last edited the form file, as specified in [MS-IPFFX] section 2.1.2.8.
Monitors: The number of monitors enabled on the client computer at the time of signing, as specified in [MS-IPFFX] section 2.1.2.12.
Width: The width of the primary monitor on the client computer at the time of signing, as specified in [MS-IPFFX] section 2.1.2.14.
Height: The height of the primary monitor on the client computer at the time of signing, as specified in [MS-IPFFX] section 2.1.2.15.
Colors: The color depth of the primary monitor on the client computer at the time of signing, as specified in [MS-IPFFX] section 2.1.2.16.
PNG: A representation of the **form view** that is displayed at the time of signing, as specified in [MS-IPFFX] section 2.1.2.20.

2.2.1.2.1.1 Protocol Server Specific Event Log

A semicolon-separated **string** of values specifying protocol server-specific information. Using the **Augmented Backus-Naur Form (ABNF)** syntax, as specified in [RFC5234], the event log MUST conform to the following specification:

```
EventLog = (Header)17(";" Entry)(";" State)7(";" Entry)
Header = 1*CHAR
Entry  = *CHAR
State  = *CHAR
```

2.2.1.2.2 Request for Add Signature Value and Context

This message is sent by the protocol client to complete the signing process.

The **message body** MUST be an **XML document**, as specified in [W3C-XML], which conforms to the following **XML schema**:

```
<xsd:element name="SignValue">
    <xsd:complexType>
        <xsd:all>
            <xsd:element ref="EventLog" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="DataDefinition" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="Value" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="Key" minOccurs="1" maxOccurs="1"/>
        </xsd:all>
    </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="EventLog" type="xsd:string"/>
<xsd:element name="DataDefinition" type="xsd:string"/>
<xsd:element name="Value" type="xsd:string"/>
<xsd:element name="Key" type="xsd:string"/>
```

EventLog: This is a semicolon-separated **string** of values containing protocol server-specific information, as specified in section 2.2.1.2.1.1.

DataDefinition: The **string** indicating the signed data block of the **form file** being signed, as specified in [MS-IPFF] section 2.2.15 and [MS-IPFF2] section 2.2.1.1.15.

Value: The value of the **digital signature**, as specified in [XMLDSig] section 4.2.

Key: The value used to populate the **X509Certificate** element of the **X509Data** node, as specified in [XMLDSig] Section 4.4.4.

2.2.1.2.3 Request for Cancel Digital Signature

This message is sent by the protocol client to cancel the signing process.

The **message body** MUST be an **XML document**, as specified in [W3C-XML], which conforms to the following **XML schema**:

```
<xsd:element name="SignCancel">
<xsd:complexType>
        <xsd:all>
            <xsd:element ref="EventLog" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="DataDefinition" minOccurs="1" maxOccurs="1"/>
        </xsd:all>
    </xsd:complexType>
</xsd:element>

<xsd:element name="EventLog" type="xsd:string"/>
<xsd:element name="DataDefinition" type="xsd:string"/>
```

EventLog: This is a semicolon-separated **string** of values containing protocol server-specific information, as specified in section 2.2.1.2.1.1.

DataDefinition: The **string** indicating the signed data block of the **form file** being signed, as specified in [MS-IPFF] section 2.2.15 and [MS-IPFF2] section 2.2.1.1.15.

2.2.2 Response Syntax

2.2.2.1 Response Status-Line

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

2.2.2.1.1 Success Response

The protocol server MUST return **HTTP OK** to indicate a success response. The response body MUST contain the detailed results specified in section 2.2.2.2.

2.2.2.1.2 Failure Response

An **HTTP** 4xx or 5xx **Status-Line**, as specified in [RFC2616] section 6.1.1. A failure response MUST only be returned by the protocol server to indicate that the request failed. There MUST NOT be a response body when a failure response is returned by the protocol server.

2.2.2.2 Response Body Syntax

A successful **HTTP OK** response returned by the protocol server MUST have a response body for any of the protocol supported functions. The response body MUST contain three lines delimited by the end-of-line indicator, **CRLF,** as specified in [RFC5234] sections 2.2 and 2.3. Each line MUST be either a **base64** encoded **string**, as specified in [RFC4648] section 4, or the **empty string**. The **strings** MUST appear in the following sequence:

- **ErrorCode**: Specified in section 2.2.2.2.1.1, section 2.2.2.2.2.1, and section 2.2.2.2.3.1.
- **ResponseData**: Specified in section 2.2.2.2.1.2, section 2.2.2.2.2.2, and section 2.2.2.2.3.2.
- **PageStateData**: Specified in section 2.2.2.2.1.3, section 2.2.2.2.2.3, and section 2.2.2.2.3.3.

The following subsections specify the individual **strings** for each protocol method.

2.2.2.2.1 Response for Retrieve Form File Hash

2.2.2.2.1.1 ErrorCode

ErrorCode is the result of processing a request for **Retrieve Form File Hash**, as specified in section 2.2.1.2.1. This value MUST be **base64** encoded, as specified in [RFC4648] section 4. The base64 encoded **string** MUST NOT be empty. When decoded, this value MUST be "1" to specify success. The decoded value MUST be a value other than 1 to indicate a failure.

2.2.2.2.1.2 ResponseData

ResponseData is the **digest** value associated with the **form file** after applying the processing rules specified in [XMLDSig] section 3. This **string** value MUST be **base64** encoded, as specified in [RFC4648] section 4.

2.2.2.2.1.3 PageStateData

PageStateData is a set of properties that the protocol server uses to **persist** implementation-specific data across protocol messages. If the protocol server persists properties, this **string** value MUST be **base64** encoded, as specified in [RFC4648] section 4. If the protocol server does not persist any properties, this value MUST be empty.

2.2.2.2.2 Response for Add Signature Value and Context

2.2.2.2.2.1 ErrorCode

ErrorCode is the result of processing a request for **Add Signature Value and Context**, as specified in section 2.2.1.2.2. The **string** value MUST be **base64** encoded, as specified in [RFC4648] section 4. The base64 encoded **string** MUST NOT be empty. When decoded, this value MUST be "1" to specify success. The decoded value MUST be a value other than 1 to indicate a failure.

2.2.2.2.2.2 ResponseData

ResponseData is a **string** value that MUST be an **empty string**.

2.2.2.2.2.3 PageStateData

PageStateData is specified in section 2.2.2.2.1.3.

2.2.2.2.3 Response for Cancel Digital Signature

2.2.2.2.3.1 ErrorCode

ErrorCode is the result of processing a request for **Cancel Digital Signature**, as specified in section 2.2.1.2.3. The **string** value MUST be **base64** encoded, as specified in [RFC4648] section 4. The base64 encoded **string** MUST NOT be empty.

2.2.2.2.3.2 ResponseData

ResponseData is a **string** value that MUST be an **empty string**.

2.2.2.2.3.3 PageStateData

PageStateData is specified in section 2.2.2.2.1.3.

3 Protocol Details

3.1 Common Details

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

3.1.1 Abstract Data Model

This section specifies a conceptual model of data organization 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 an implementation adhere to this conceptual model as long as the implemented model's external behavior is consistent with the behavior specified in this document.

The protocol server maintains a mapping between the client data and the form file persisted on the server. This mapping can be stored as part of the **EventLog**, as specified in section 2.2.1.2.1.1, section 2.2.1.2.2, and section 2.2.1.2.3.

The protocol server accepts requests to create and add a **digital signature** to the form file it maintains. This multi-step process, known as a signing session, requires the protocol server to modify the form file using data sent by the protocol client. A form file can have three different states during a signing session:

- **Initial State**: The state of the form file before a signing session is started.
- **Pre-Signed State**: The state of the form file after the protocol server successfully processed a **Retrieve Form File Hash** message, as specified in section 2.2.1.2.1.
- **Signed Complete State**: The state of the form file after the protocol server has successfully processed an **Add Signature Value and Context** message, as specified in section 2.2.1.2.2.

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 Protocol Client Details

3.2.1 Abstract Data Model

The abstract data model is 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

Request messages sent by the protocol client MUST be in the syntax specified in section 2.2.1.

The protocol client MUST send request messages to the protocol server in one of the valid request sequences:

- A request for **Retrieve Form File Hash** followed by a request for **Add Signature Value and Context**.
- A request for **Retrieve Form File Hash** followed by a request for **Cancel Digital Signature**.

Any other request sequence is invalid and MUST NOT be used while sending requests to the protocol server.

For processing the response for **Cancel Digital Signature**, the protocol client MUST ignore the **ErrorCode** value returned by the protocol server.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Protocol Server Details

3.3.1 Abstract Data Model

The abstract data model is 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

When processing the messages sent by the protocol client, the protocol server modifies the state of the **form file**, as specified in section 3.1.1.

The protocol server MUST run the sequencing rules specified in section 3.3.5.1 if the protocol client **message body** contains the **XML** specified in section 2.2.1.2.1.

The protocol server MUST run the sequencing rules specified in section 3.3.5.2 if the protocol client message body contains the XML specified in section 2.2.1.2.2.

The protocol server MUST run the sequencing rules specified in section 3.3.5.3 if the protocol client message body contains the XML specified in section 2.2.1.2.3.

3.3.5.1 Process the request for Retrieve Form File Hash

The protocol server MUST process this message as follows:

- If an error occurs while processing this request, the protocol server MUST stop processing this request and MUST return an **ErrorCode** other than 1, as specified in section 2.2.2.2.1.1. When an error occurs while processing the request, the protocol server MUST set the state of the **form file** to **Initial State**.
- The protocol server MUST add the **Signature** element, as specified in [XMLDSig] section 2, to the form file identified by the protocol client message.
- The protocol server MUST use the protocol message data, as specified in section 2.2.1.2.1, to populate the **digital signature** properties specified in [MS-IPFFX] section 2.1.2.
- The protocol server MUST determine the **digest** value associated with the form file by applying the processing rules specified in [XMLDSig] section 3.0.
- If the operations specified in this section are successful, the protocol server MUST move the state of the form file to **Pre-Signed State**, as specified in section 3.1.1.
- The protocol server MUST send a response, as specified in section 2.2.2.2.1, containing:
  - **ErrorCode**, as specified in section 2.2.2.2.1.1.
  - **ResponseData** as the digest value associated with the form file in the case of success, or empty if the **ErrorCode**, as specified in section 2.2.2.2.1.1, indicates a failure.
  - **PageStateData** as a set of properties of protocol server implementation specific data, as specified in section 2.2.2.2.1.3.

3.3.5.2 Process the request for Add Signature Value and Context

The protocol server MUST process this message as follows:

- If an error occurs while processing this request, the protocol server MUST stop processing this request and MUST return an **ErrorCode** other than 1, as specified in section 2.2.2.2.2.1.

- If the **State** entry of the **EventLog** in the client message, as specified in section 2.2.1.2.1.1, is different from the **PageStateData** sent in the response for **Retrieve Form File Hash**, as specified in section 2.2.2.2.1, the protocol server MUST stop processing this request and SHOULD<1> return an **ErrorCode** other than 1.
- The protocol server MUST verify that the value of **DataDefinition** sent by the protocol client message, as specified in section 2.2.1.2.2, is identical to the value received in the request for **Retrieve Form File Hash**, as specified in section 2.2.1.2.1. For any other values of the **DataDefinition** property, the protocol server SHOULD<2> return an **ErrorCode** other than 1, as specified in section 2.2.2.2.2.1.
- The protocol server MUST update the **SignatureValue** element specified by [XMLDSig] section 4.2 and the **X509Data** element specified by [XMLDSig] section 4.4.4.
- If the operations specified in this section are successful, the protocol server MUST move the state of the **form file** to **Signed Complete State**, as specified in section 3.1.1.
- After moving the state of the form file to **Signed Complete State**, the protocol server MUST move the state of the form file to **Initial State**.
- The protocol server MUST send a response, as specified in section 2.2.2.2.2, containing the following:
  - **ErrorCode**, as specified in section 2.2.2.2.2.1.
  - ResponseData as an **empty string**.
  - **PageStateData** as a set of properties of protocol server implementation-specific data, as specified in section 2.2.2.2.1.3.

3.3.5.3 Process the request for Cancel Digital Signature

The protocol server MUST process the message as follows:

- If an error occurs while processing this request, the protocol server MUST stop processing this request and MUST return an **ErrorCode** other than 1, as specified in section 2.2.2.2.3.1.
- If the **State** entry of the **EventLog** in the client message, as specified in section 2.2.1.2.1.1, is different from the **PageStateData** sent in the response for **Retrieve Form File Hash**, as specified in section 2.2.2.2.1, the protocol server SHOULD<3> return an **ErrorCode** other than 1.
- The protocol server MUST verify that the value of the **DataDefinition** sent by the protocol client message, as specified in section 2.2.1.2.3, is identical to the one received in the request for **Retrieve Form File Hash**, as specified in section 2.2.1.2.1. For any other values of the **DataDefinition** property, the protocol server SHOULD<4> return an **ErrorCode** other than 1.
- The protocol server MUST remove any **digital signature** elements that have been added to the **form file** as the result of processing a request for a **Retrieve Form File Hash** message in the same signing session.
- If the operations specified in this section are successful, the protocol server MUST move the state of the form file to **Initial State**, as specified in section 3.1.1.
- The protocol server MUST send a response, as specified in section 2.2.2.2.3, containing the following:
  - **ErrorCode**, as specified in section 2.2.2.2.3.1.
  - **ResponseData** as an **empty string**.

- **PageStateData** as a set of properties of protocol server implementation-specific data, as specified in section 2.2.2.2.1.3.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

4 Protocol Examples

The examples in this section show the messages exchanged when a protocol client makes a successful **HTTP** request to a protocol server using this protocol.

4.1 Messages for Retrieve Form File Hash

The following subsections show the interaction between the protocol client and protocol server to initiate a signing process.

4.1.1 Request Body

The following example is a protocol client request message to initiate a signing process, as specified in section 2.2.1.2.1.

```
POST /_layouts/Signature.FormServer.aspx HTTP/1.1
Accept: */*
Content-Type: text/xml
Host: contoso

<SignRequest><EventLog>1
8;5;872b5f4e78e1493582a00f161142fe62_be2940ab07fd4251bbcdf0041d2bc5da;AHLZ4GUVVKKSISFM2IIBTIR
CKQRSCL2TJFKEKUZPKY2C6RCTJFDS6RSPKJGVGL2UIVGVATCBKRCS4WCTJYVGQOCRKNJG4STIHFLVE6BUKVTE6N2CJ54W
OSZYGNLWQT3QPBUUMUBYM5GEMQSIGY3EQ4Y;0;;%2Fsites%2Fv4%2Fdsig%2Fforms%2Ftemplate.xsn;http%3A%2F
%contoso%2Fsites%2Fv4%2Fdsig%2Fforms%2Ftemplate.xsn;;http%3A%2F%2Fcontoso%2Fsites%2Fv4;;1;1;0
;1;0;0;633997851883411000;;FormControl;0;1033;1033;0;13;Op0ZEFdM7QYQwAg6724FORnJu727/iMxjDBCQ
U2UHqX8B/P+trBObL+1XnQzZ/s068f9gUYZp1YxHRgsxi3A0w==|633997856567555876</EventLog><DataDefinit
ion>group1</DataDefinition><Comment></Comment><Time>2010-01-
22T19:34:33Z</Time><OS>6.0</OS><Browser>Microsoft Internet Explorer
7.0</Browser><Version>14</Version><Monitors>1</Monitors><Width>1404</Width><Height>1126</Heig
ht><Colors>16</Colors><PNG>(PNG IMAGE)</PNG></SignRequest>
```

4.1.2 Response Body

The following example is a protocol server response message to a request to initiate a signing process, as specified in section 2.2.2.2.1. The **string** value of **PageStateData**, as specified in section 2.2.2.2, is empty in the message example. The required CRLF characters are present.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding

AQ==
RLVG1nrhjkgeTBUNEGE1/cQoMlo=
```

4.2 Messages for Add Signature Value and Context

The examples in the following subsections show the interaction between the protocol client and protocol server to add a signature value and context to a **form file**.

4.2.1 Request Body

The following example is a protocol client request message to add a signature value and context to a **form file**, as specified in section 2.2.1.2.2. The **CERTIFICATE CONTEXT** in the following example is a binary representation of a **certificate** specified as **key** in section 2.2.1.2.2. The value of **CERTIFICATE CONTEXT** is abbreviated for readability.

```
POST /_layouts/Signature.FormServer.aspx HTTP/1.1
Content-Type: text/xml
Host: contoso

<SignValue><EventLog>1
8;5;872b5f4e78e1493582a00f161142fe62 be2940ab07fd4251bbcdf0041d2bc5da;AHLZ4GUVVKKSISFM2IIBTIR
CKQRSCL2TJFKEKUZPKY2C6RCTJFDS6RSPKJGVGL2UIVGVATCBKRCS4WCTJYVGQOCRKNJG4STIHFLVE6BUKVTE6N2CJ54W
OSZYGNLWQT3QPBUUMUBYM5GEMQSIGY3EQ4Y;0;;%2Fsites%2Fv4%2Fdsig%2Fforms%2Ftemplate.xsn;http%3A%2F
%2Fcontoso%2Fsites%2Fv4%2Fdsig%2Fforms%2Ftemplate.xsn;;http%3A%2F%2Fcontoso%2Fsites%2Fv4;;1;1
;0;1;0;0;633997851883411000;;FormControl;0;1033;1033;0;13;Op0ZEFdM7QYQwAg6724FORnJu727/iMxjDB
CQU2UHqX8B/P+trBObL+1XnQzZ/s068f9gUYZp1YxHRgsxi3A0w==|633997856567555876</EventLog><DataDefin
ition>group1</DataDefinition><Value>iWFKsksj4iFefBoOd1FCLLDs7B8vfqHd7s2IJaI6r2r0kL0HR1Zq5Q33v
AuNZYsNYUuiS4ZeCxznY69BY5eJ1SdnrQlOTDMIldFvDFCF7H+mPIRYQlkTZZiTYZInmIhUkFVh4q+/mbBVRTT5xfFM5Q
rOD41R0jLPsrLCNEk=</Value><Key>(CERTIFICATE CONTEXT)</Key></SignValue>
```

4.2.2 Response Body

The following example is a protocol server response to add a signature value and context to a **form file**, as specified in section 2.2.2.2.2. The **string** values of **ResponseData** and **PageStateData**, as specified in section 2.2.2.2, are empty in the message example. The required CRLF characters are present.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
Content-Length: 6

AQ==
```

4.3 Messages for Request for Cancel Digital Signature

The following subsection contains an example that shows the interaction between the protocol client and protocol server to cancel the application of a **digital signature** to a **form file**.

4.3.1 Request Body

The following example is a protocol client request to cancel the application of a **digital signature** to a **form file**, as specified in section 2.2.1.2.3.

```
POST /_layouts/Signature.FormServer.aspx HTTP/1.1
Content-Type: text/xml
Host: contoso

<SignCancel><EventLog>1
8;5;872b5f4e78e1493582a00f161142fe62 be2940ab07fd4251bbcdf0041d2bc5da;AHLZ4GUVVKKSISFM2IIBTIR
CKQRSCL2TJFKEKUZPKY2C6RCTJFDS6RSPKJGVGL2UIVGVATCBKRCS4WCTJYVGQOCRKNJG4STIHFLVE6BUKVTE6N2CJ54W
OSZYGNLWQT3QPBUUMUBYM5GEMQSIGY3EQ4Y;0;;%2Fsites%2Fv4%2Fdsig%2Fforms%2Ftemplate.xsn;http%3A%2F
%2Fcontoso%2Fsites%2Fv4%2Fdsig%2Fforms%2Ftemplate.xsn;;http%3A%2F%2Fcontoso%2Fsites%2Fv4;;1;1
;0;1;0;0;633997851883411000;;FormControl;0;1033;1033;0;13;Op0ZEFdM7QYQwAg6724FORnJu727/iMxjDB
CQU2UHqX8B/P+trBObL+1XnQzZ/s068f9gUYZp1YxHRgsxi3A0w==|633997856567555876</EventLog><DataDefin
ition>group1</DataDefinition></SignCancel>
```

4.3.2 Response Body

The following example is a protocol server response to cancel the application of a **digital signature** to a **form file**, as specified in section 2.2.2.2.3. The string values of **ResponseData** and **PageStateData**, as specified in section 2.2.2.2, are empty in the message example. The required CRLF characters are present.

```
HTTP/1.1 200 OK
```

```
Cache-Control: private
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
Content-Length: 6

Ag==
```

5 Security

5.1 Security Considerations for Implementers

The protocol server and protocol client both encrypt data stored on the protocol server when applying a **digital signature** to a **form file**. A digital signature (2) is only as secure as the algorithms used to generate its encrypted elements. If an algorithm used to generate encrypted elements of the digital signature (2) is compromised, the integrity of the digital signature (2) can no longer be verified.

Security considerations for digitally signed form files can be found in [MS-IPFFX] section 4.1.

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 updates to those products.

- Microsoft InfoPath 2010
- Microsoft InfoPath 2013
- Microsoft SharePoint Server 2010
- Microsoft SharePoint Server 2013
- Microsoft SharePoint Server 2016
- Microsoft SharePoint Server 2019

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates 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 3.3.5.2: SharePoint Server 2010 returns an **HTTP OK** response with a text/html response body containing an error message.

<2> Section 3.3.5.2: SharePoint Server 2010 returns an HTTP OK response with a text/html response body containing an error message.

<3> Section 3.3.5.3: SharePoint Server 2010 returns an HTTP OK response with a text/html response body containing an error message.

<4> Section 3.3.5.3: SharePoint Server 2010 returns an HTTP OK response with a text/html response body containing an error message.

7 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

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.
- A document revision that captures changes to protocol functionality.

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 **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

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

Section	Description	Revision class
6 Appendix A: Product Behavior	Updated list of supported products.	Major

8 Index

P

R

S

T

V