[MS-IPDSP]:
InfoPath Digital Signing 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.

Copyright © 2010 Microsoft Corporation.

Release: Sunday, December 19, 2010

Revision Summary

Date	Revision History	Revision Class	Comments
02/19/2010	1.0	Major	Initial Availability
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

Copyright © 2010 Microsoft Corporation.

Copyright © 2010 Microsoft Corporation.

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.

1.1 Glossary

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

Augmented Backus-Naur Form (ABNF)
authentication
certificate
ciphertext
HTTP OK
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)

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

digest
digital signature
form file
form view
Request-URI
site
Status-Line
URI (Uniform Resource Identifier)
URL (Uniform Resource Locator)
XML Schema

The following terms are specific to this document:

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

1.2 References

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", June 2008.

[MS-IPFF2] Microsoft Corporation, "InfoPath Form Template Format Version 2", July 2009.

[MS-IPFFX] Microsoft Corporation, "InfoPath Form File Format Specification", June 2008.

[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

Copyright © 2010 Microsoft Corporation.

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

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006, http://www.ietf.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.ietf.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., LaMacchia, B., and Simon, E., "XML-Signature Syntax and Processing", February 2002, http://www.w3.org/TR/xmldsig-core/

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary", June 2008.

1.3 Protocol Overview (Synopsis)

This protocol enables a protocol client to communicate with a protocol server over an **HTTP** connection to apply a **digital signature (2)** to a **form file** that is stored on the protocol server. To apply a digital signature (2) 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 (2) to a form file. Using this protocol function, the protocol client sends a request to the protocol server which 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 (1)** 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 (2). The application of the digital signature (2) 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.

Copyright © 2010 Microsoft Corporation.

1.4 Relationship to Other Protocols

This protocol requires the HTTP/1.0 [RFC1945] or HTTP/1.1 [RFC2616] protocol for message transport. The protocol transmits messages using the HTTP protocol as specified in [RFC2616] or the **HTTPS** protocol as specified in [RFC2818].

The following figure shows the transport stack that this protocol uses:

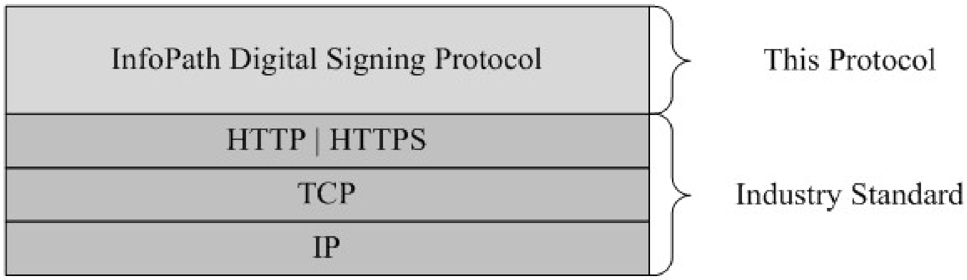

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

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

- This protocol operates against a **site (2)** identified by a **URL** known by the protocol client.
- The data represented by the **Protocol Server Specific Event Log**, as specified in section 2.2.1.2.1.1, are known by the protocol client.
- **DataDefinition**, as specified in section 2.2.1.2.1 is known by the protocol client.
- **PageStateData**, as specified 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 (2) 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

This document covers versioning issues in the following areas:

- Supported Transports: This protocol uses multiple transports with HTTP as specified 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(2) as specified in [MS-IPFFX] section 2.1.2.1.

Time: The system date and time for the client computer as 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 **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 (2) 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.

Copyright © 2010 Microsoft Corporation.

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 detailed results as 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, 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]. Each line MUST be either a Base64 encoded string, as specified in [RFC4648], 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

Specifies 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]. The Base64 encoded string MUST NOT be empty. When decoded, this value MUST be 1 to specify success. Any other decoded value indicates a failure.

2.2.2.2.1.2 ResponseData

Specifies 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] and MUST NOT be empty.

Copyright © 2010 Microsoft Corporation.

2.2.2.2.1.3 PageStateData

Specifies a set of properties that the protocol server uses to persist implementation specific data across protocol messages. This string value MUST be Base64 encoded, as specified in [RFC4648]. 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

Specifies 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]. The Base64 encoded string MUST NOT be empty. When decoded, this value MUST be 1 to specify success. Any other decoded value indicates a failure.

2.2.2.2.2.2 ResponseData

This string value MUST be an empty string.

2.2.2.2.2.3 PageStateData

As 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

Specifies 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]. The Base64 encoded string MUST NOT be empty. When decoded, this value MUST be 1 to specify success. Any other decoded value indicates a failure.

2.2.2.2.3.2 ResponseData

This string value MUST be an empty string.

2.2.2.2.3.3 PageStateData

As specified in section 2.2.2.2.1.3.

2.3 Directory Service Schema Elements

None.

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. An implemented model's external behavior should be 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 (2) 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. The signed complete state is equivalent to the initial state.

If the protocol client cancels the process, the protocol server MUST restore the form file to the **Initial State**.

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.

Copyright © 2010 Microsoft Corporation.

3.1.7 Other Local Events

None.

3.2 Protocol 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

None.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Protocol 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.

Copyright © 2010 Microsoft Corporation.

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 as 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 as 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 as 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 specified in section 2.2.1.2.1 as follows:

- The protocol server MUST verify that the form file identified by the protocol client message is in the **Initial State** as specified in section 3.1.1.
- If the form file is in any state other than **Initial State**, or 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 MUST set the state of the form file to the **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 (2) 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, then 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. If the protocol server does not persist any properties this value MUST be empty.

3.3.5.2 Process the request for Add Signature Value and Context

The protocol server MUST process this message, as specified in section 2.2.1.2.2 as follows:

The protocol server MUST verify that the form file identified by the protocol client message is in the **Pre-Signed State** as specified in section 3.1.1.

If the form file is in any state other than **Pre-Signed State**, or 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 than 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 MUST 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 2.2.1.2.2, is identical to the value received in the request for **Retrieve Form File Hash** as specified in section 2.2.1.2.1. The protocol server MUST return an **ErrorCode**, as specified in section 2.2.2.2.2.1, indicating a signing error for any other values of the **DataDefinition** property.

The protocol server MUST update the **SignatureValue** element specified by [XMLDSig] section 4.2 and X509Data element specified by [XMLDSig] section 4.4.4.

If the operations specified in this section are successful, then the protocol server MUST move the state of the form file to the **Signed Complete State**, as specified in section 3.1.1.

The protocol server MUST send a response, as specified in section 2.2.2.2.2, containing:

- **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. If the protocol server does not persist any properties this value MUST be empty.

3.3.5.3 Process the request for Cancel Digital Signature

The protocol server MUST process the message, as specified in section 2.2.1.2.3 as follows:

The protocol server MUST verify that the form file identified by the protocol client message is in the **Pre-Signed State**, as specified in section 3.1.1.

If the form file is in any state other than the **Pre-Signed State**, or 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 than the **PageStateData** sent in the response for **Retrieve Form File Hash**, as specified in section 2.2.2.2.1, the protocol server MUST 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. The protocol server MUST return an **ErrorCode** indicating a signing error for any other values of the **DataDefinition** property.

The protocol server MUST remove any digital signature (2) elements that have been added to the form file as the result of processing a request for **Retrieve Form File Hash** message in the same signing session.

If the operations specified in this section are successful, then 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:

- **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. If the protocol server does not persist any properties this value MUST be empty.

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 Messages for Retrieve Form File Hash

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

4.1.1 Request Body

An example of a protocol client request message to initiate a signing process, as specified in section 2.2.1.2.1, is illustrated here:

```
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

An example of a protocol server response message to a request to initiate a signing process, as specified in section 2.2.2.2.1, is illustrated in this section. 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 in the message example.

```
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 following subsections illustrate the interaction between the protocol client and protocol server to add a signature value and context to a form file.

Copyright © 2010 Microsoft Corporation.

4.2.1 Request Body

An example of 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, is illustrated here. The **CERTIFICATE CONTEXT** specified in the following example is a binary representation of a certificate specified as **key** in the 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

An example of a protocol server response to add a signature value and context to a form file, as specified in section 2.2.2.2.2, is illustrated in this section. 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 in the message example.

```
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 illustrates 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

An example of 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, is illustrated here:

```
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

An example of 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, is illustrated in this section. 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 in the message example.

```
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 is only as secure as the algorithms used to generate its encrypted elements. If an algorithm used to generate encrypted elements of the digital is compromised, then the integrity of the digital signature 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 released service packs:

- Microsoft® InfoPath® 2010
- 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.

Copyright © 2010 Microsoft Corporation.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

Copyright © 2010 Microsoft Corporation.

Copyright © 2010 Microsoft Corporation.